Modélisation des systèmes d'information

INFO-MC-OMGL2

S. LEPREUX

Cours adapté du cours de Mme Grislin – MC-OMGL2 2007-2008

Références bibliographiques

- MULLER P.A. "Modélisation objet avec UML", Eyrolles, 1997.
- FANNADER R., LEROUX H. "UML Principes de modélisation", Dunod, 1999.
- ROQUES P., VALLE F. "UML en action", Eyrolles, 2000.
- ROQUES P. « UML 2 par la pratique », Eyrolles, 2001, 2006.
- DEBRAUWER L., VAN DER HEYDE F. « UML 2. Initiation, exemples et exercices corrigés », Eni ed., 2005.
- PILONE D., PITMAN, N. « UML 2 en concentré » O'Reilly, 2006.
- OMG " Unified Modeling Language Specification - version 1.4", sept. 2001.

Internet

Les sites suivants contiennent de la documentation sur UML :

- www.celigent.com/omg/umlrtf : UML 1.4 RTF: OMG UML
- www.celigent.com/omg/umlrtf/tutorials.htm : tutoriels
- www.omg.org/uml/ site de l'OMG
- http://uml.free.fr : doc. en français
- http://www.coget.com/uml
- http://www.objektteknik.se/uml/
- http://www.rational.com/ produit AGL UML : Rose
- http://argouml.tigris.org : produit AGL : Argo/UML freeware
- http://bouml.free.fr/ : produit AGL UML2
- http://uml.netbeans.org/ : produit AGL UML

Plan (12 h de cours)

- Introduction à UML.
  Les méthodes orientées Objet.
- Les cas d'utilisation
- Diagrammes d'objets et de classes
- Diagramme de séquence
- Diagrammes d'états et d'activité
- Diagramme de communication
- Diagrammes de composants et de déploiement
- Génération de code. Outils. Tests logiciels

Plan (12 h de cours)

- Introduction à UML.
  Les méthodes orientées Objet.
- Les cas d'utilisation
- Diagrammes d'objets et de classes
- Diagramme de séquence ← **Interrogation (prévue)**
- Diagrammes d'états et d'activité
- Diagramme de communication
- Diagrammes de composants et de déploiement
- Génération de code. Outils. Tests logiciels

← **DS**

Introduction

- **Génie Logiciel** = ensemble de méthodes, modèles et outils pour construire des logiciels.

- Pour développer un logiciel, il faut :
  - Déterminer les besoins : *quel est le cahier des charges ?*
  - Analyse : *que fera le logiciel ?*
  - Conception : *comment le fera-t-il ?*
  - Implémentation : *avec quel code ?*
  - Test : *le logiciel sera-t-il conforme aux besoins ?*
  - Déploiement : *sur quels matériels ?*

Méthode ≠ Modèle

- **Méthode de Génie Logiciel** = organisation des activités précédentes : "mode d'emploi" décrivant les phases de développement
- **Modèle** = description abstraite du système : pour préciser certaines caractéristiques

  Le modèle doit être :
  - Support à la conception par validation progressive du modèle
    - Communication entre concepteurs
    - Communication entre concepteurs et utilisateurs
  - Représenté par un ensemble de documents : diagrammes, textes, ...
- *Rem. : Une méthode fournit "un plan", une chronologie d'utilisation de plusieurs modèles.*

- Standard d'écriture des différents modèles : UML

iut VALENCIENNES CAMBRAI - MAUBEUGE
depuis 1967
PLUS DE 40 ans D'EXPERIENCE
Informatique DUT INFO

Besoins en développement logiciel

- Les logiciels sont de plus en plus grands et complexes

- Souci d'efficacité :
  - Ne pas "réinventer la roue" à chaque développement ==> REUTILISER !
  - Pouvoir maintenir facilement le logiciel :
    - ajouter des fonctionnalités,
    - s'adapter à de nouvelles situations...

Qu'est-ce qu'un objet ?

- **objet informatique** = représentation abstraite des entités d'un monde réel ou virtuel, dans le but de les piloter ou de les simuler

- Un objet est formé d'un état et d'un comportement
  - Etat de l'objet : l'ensemble des valeurs de ses caractéristiques à un instant donné
    - représenté par des **attributs**
      Partie statique de l'objet
  - Comportement de l'objet : manière dont l'objet agit et réagit
    - représenté par des **méthodes** ou **opérations**
      Partie dynamique de l'objet

Historique des objets

1967 : langage de programmation Simula

années 70 : langage de programmation SMALLTALK

années 80 : fondements théoriques

C++, Objective C, ...

années 90 : méthodes d'analyse et de conception OO (Booch, OMT, ROOM, Fusion, HOOD, Catalysis, ...), Java

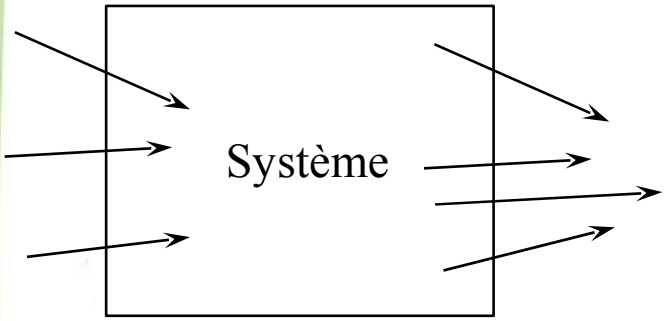
1997 : UML standardisé par l'OMG

(Object Management Group)

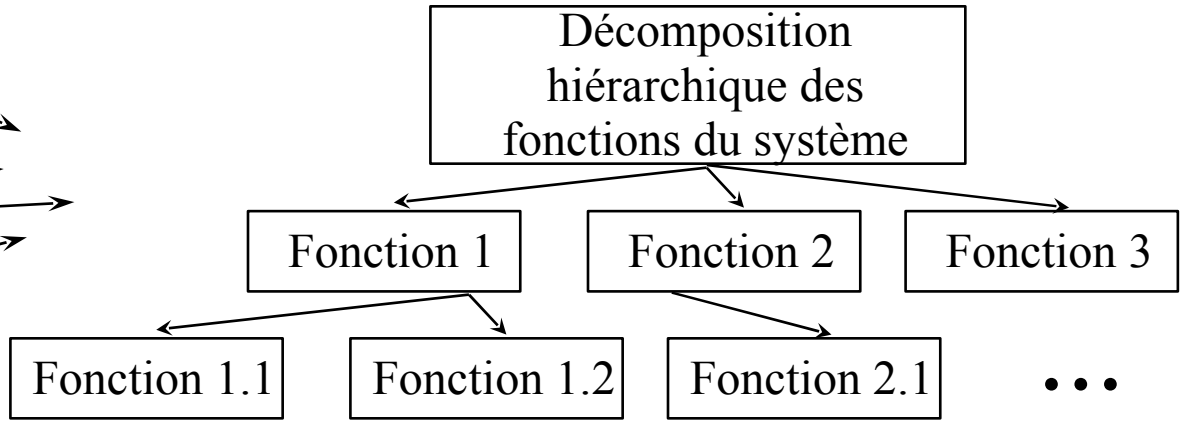
2005 : UML2

Principe de l'Analyse Fonctionnelle

- Analyse fonctionnelle :
  - accent mis sur ce que fait le système (ses fonctions)
  - identification des fonctions du système puis décomposition en sous-fonctions, récursivement, jusqu'à obtention de fonction élémentaires, implémentables directement
  - la fonction détermine la structure

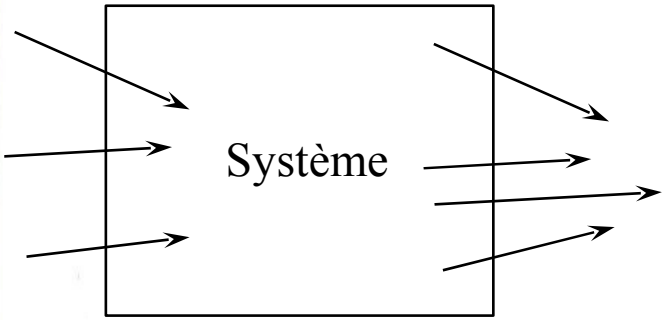


«Que fait le système ?»

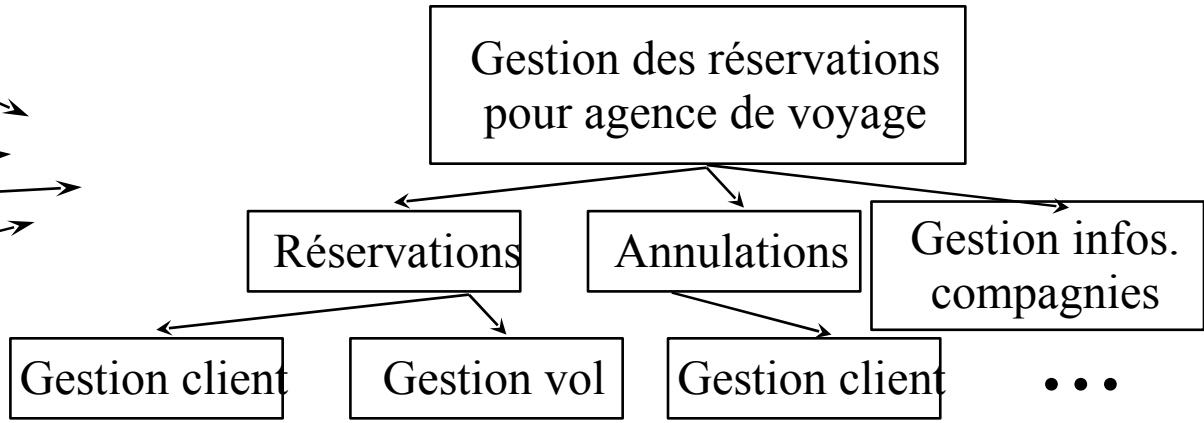


Principe de l'Analyse Fonctionnelle

- Analyse fonctionnelle :
  - accent mis sur ce que fait le système (ses fonctions)
  - identification des fonctions du système puis décomposition en sous-fonctions, récursivement, jusqu'à obtention de fonction élémentaires, implémentables directement
  - la fonction détermine la structure

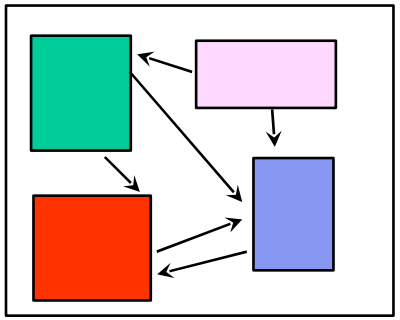


«Que fait le système ?»

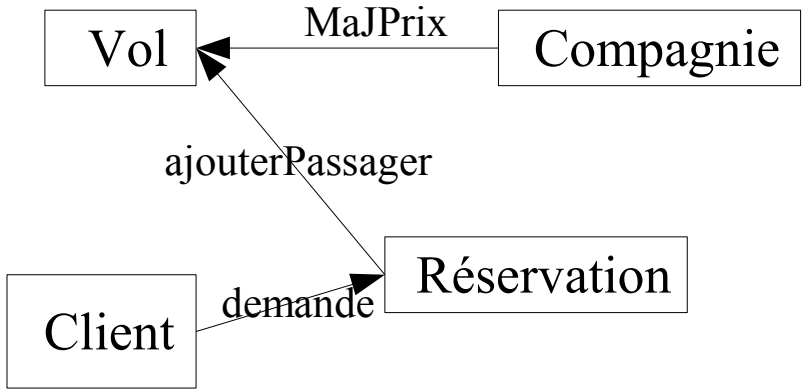


Principe de l'Analyse Orientée Objet

- Analyse OO :
  - accent mis sur ce qu'est le système (ses composants) : Un système informatique est vu comme un ensemble structuré d'éléments qui collaborent
  - identification des composants du système : les objets
  - Méhode (appel de) = collaboration entre objets
  - les méthodes sont indépendantes de la structure
  - un objet intègre à la fois des données (attributs) et des opérations (méthodes)



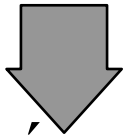
«De quoi se compose le système ?»



Principes des méthodes OO

Constat :

les opérations et les données sont étroitement liées



ne pas les séparer lors de la conception

conception « classique » (AP1 : découpage fonctionnel)

Difficulté :

déterminer les objets fondamentaux du système

PLUS DE 40 ans D'EXPERIENCE depuis 1967
iut VALENCIENNES CAMBRAI • MAUBEUGE
Informatique DUT INFO

Principes des méthodes OO

- Analyse OO :
  - trouver les objets
  - les organiser
  - décrire leurs interactions
  - définir leurs opérations (méthodes)
  - définir l'intérieur des objets
- Programmation OO
- Test OO

IUT

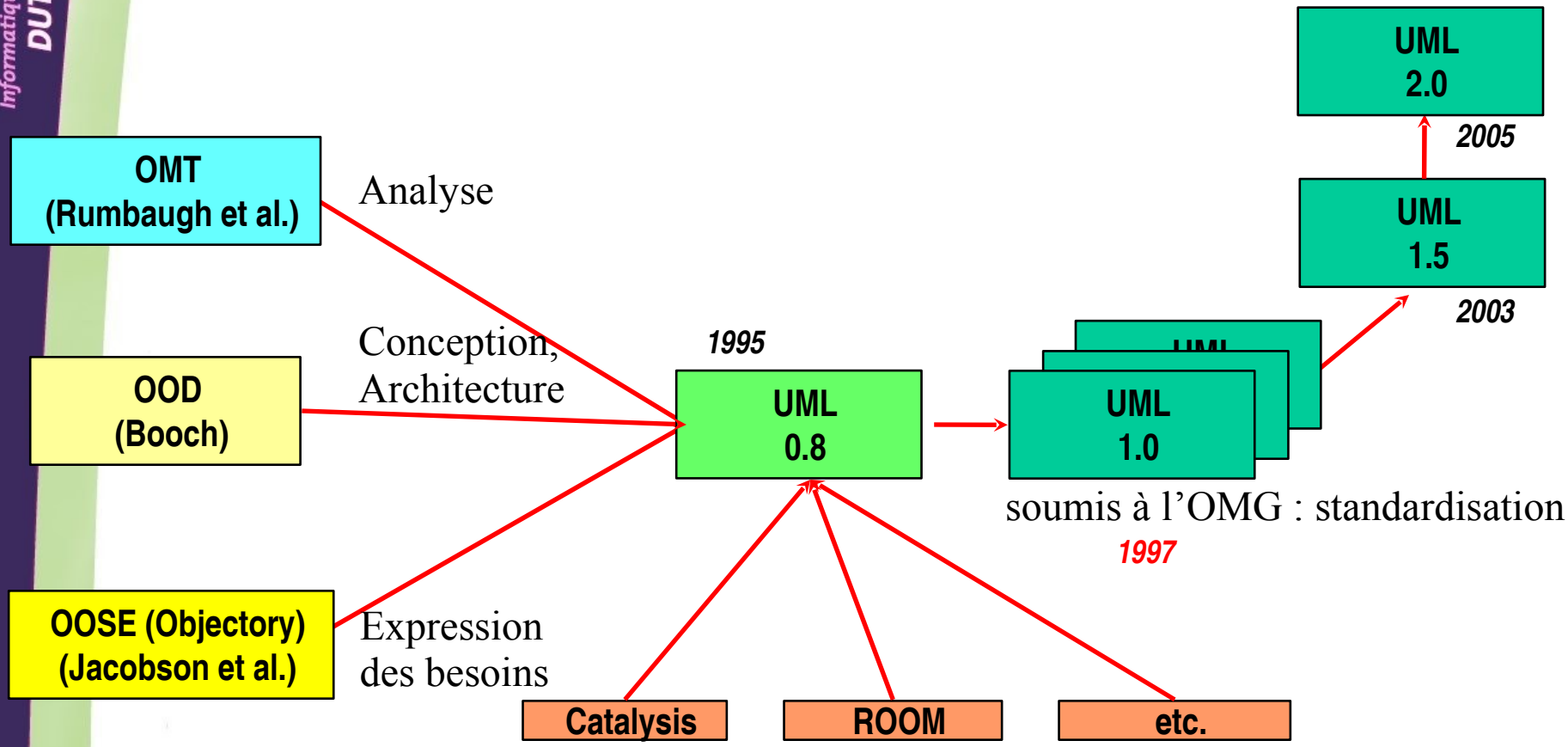
iut VALENCIENNES CAMBRAI - MAUBEUGE
depuis 1967
PLUS DE 40 ans D'EXPERIENCE
Informatique DUT INFO

Qu'est-ce que UML ?

UML = **U**nified **M**odeling **L**anguage
for Object-Oriented Development

- 1er standard international en conception de système d'information.
- provient de l'unification de différents modèles Orientés-Objet.
- adapté et utilisé pour la conception orientée objet

Convergence



Notations à l'origine de UML

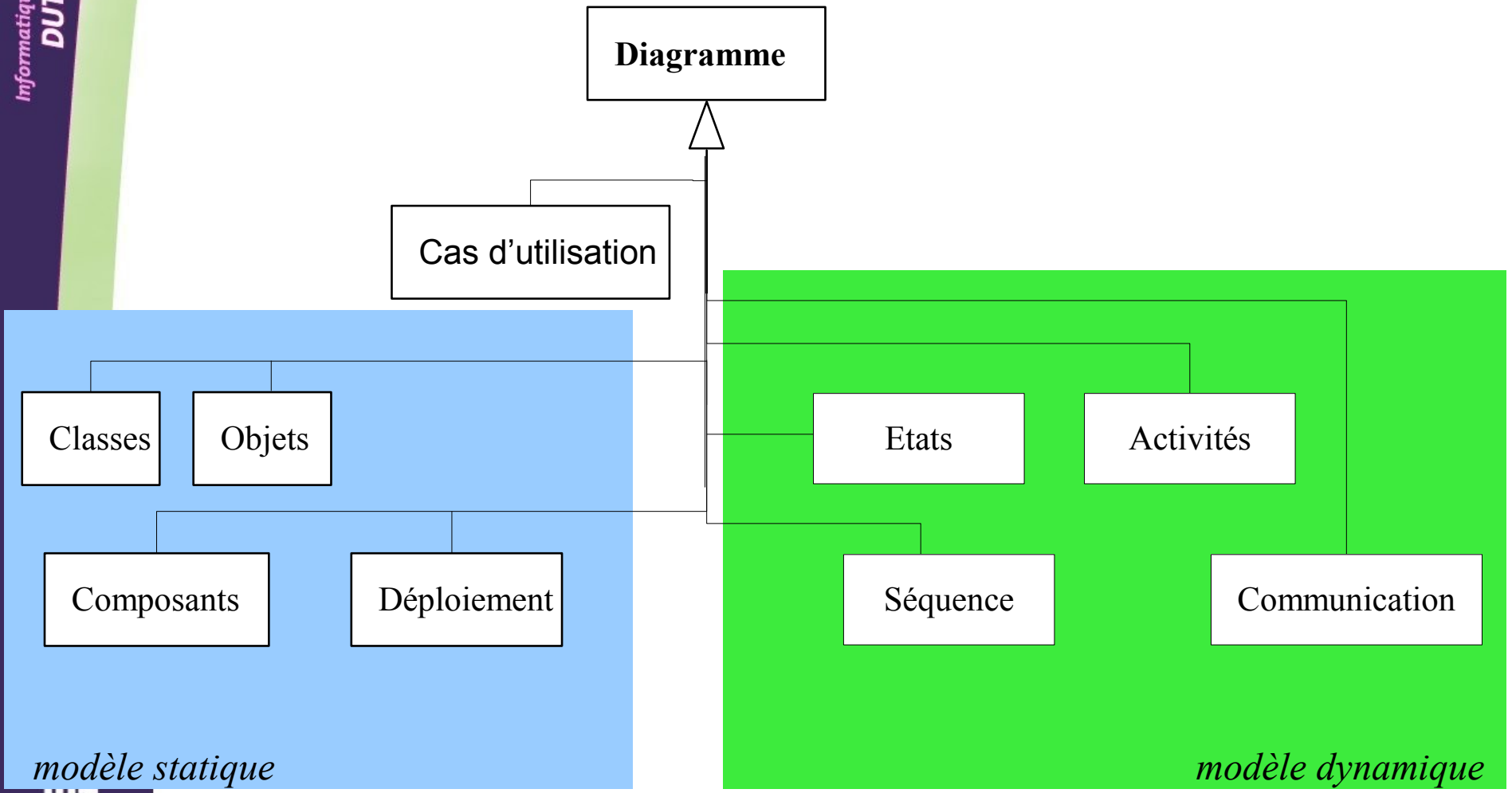
- **OMT** (Object Modeling Technique)
  de James **RUMBAUGH et al.** (1991)
- **OOD** (Object-Oriented Design)
  de Grady **BOOCH**, pionnier de l'Orienté-Objet (1981)
  Version la plus aboutie : **Booch'93**
- **OOSE** (Object-Oriented Software Engineering)
  = **Objectory** de Ivar **JACOBSON**

Caractéristiques de UML

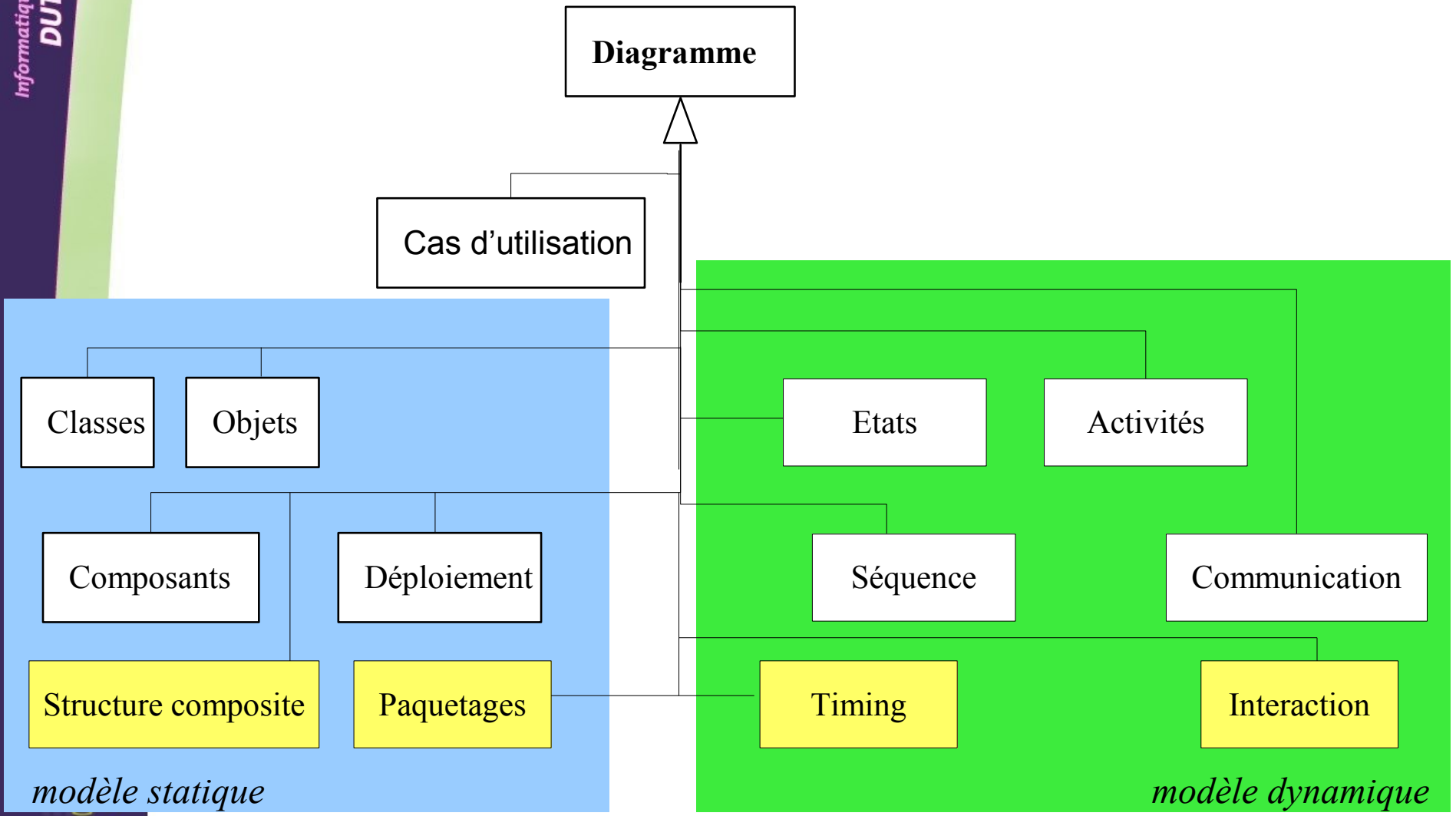
- **UML ne standardise pas le processus de développement.**
- UML est un **langage**, une notation essentiellement graphique.

- Les *blocs de base* d'UML sont :
  - les éléments de modélisation (classes, interfaces, composants, use cases, etc.)
  - les relations entre éléments (associations, généralisation, dépendences, etc.)
  - les diagrammes (class diagrams, use case diagrams, interaction diagrams, etc.)

Les diagrammes d'UML 1.4



Les diagrammes d'UML 2.0



UML : méthode ou modèle ?

Méthode = langage + processus

- **langage de modélisation** : notation utilisée pour décrire les éléments de modélisation
- **processus** : décrit les étapes et les tâches à effectuer pour mener à bien la conception

processus $\not\subset$ UML, donc :

UML n'est pas une *méthode*

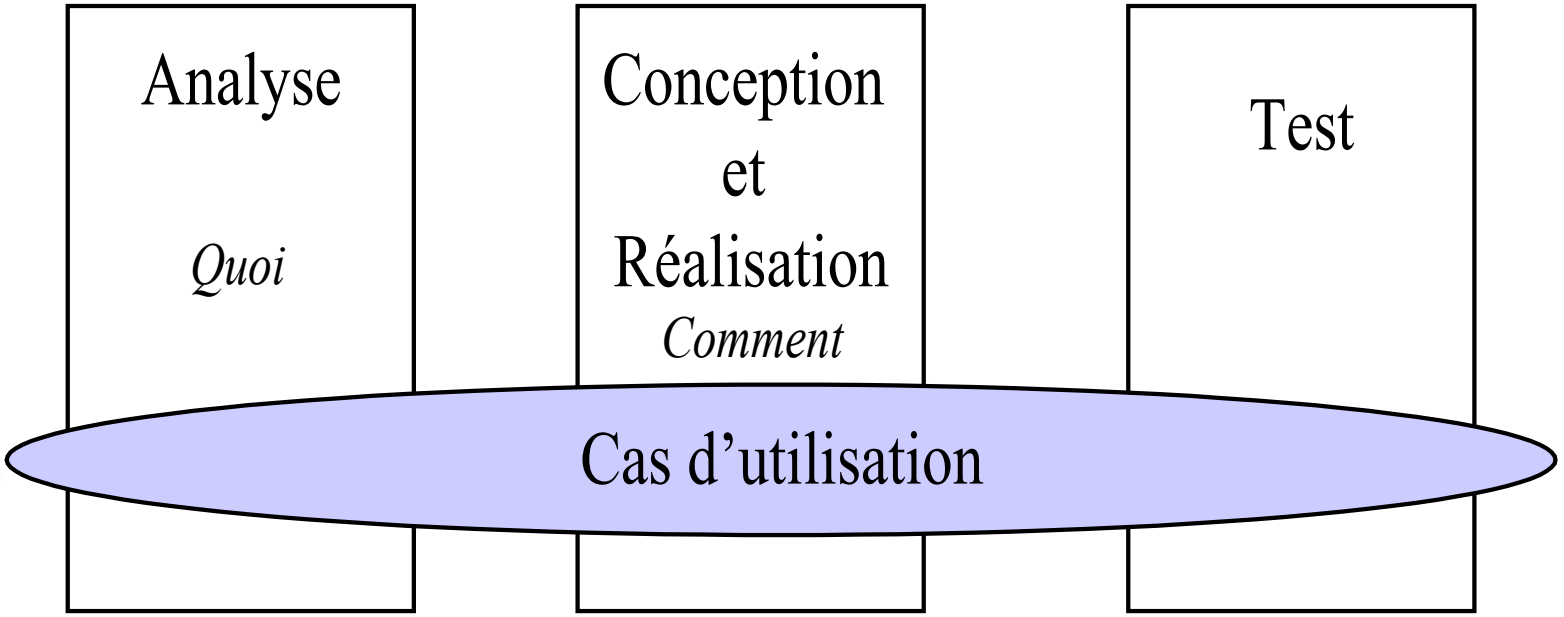
mais

un *langage* !

UML = des modèles => une méthode

un processus est préconisé : le processus unifié

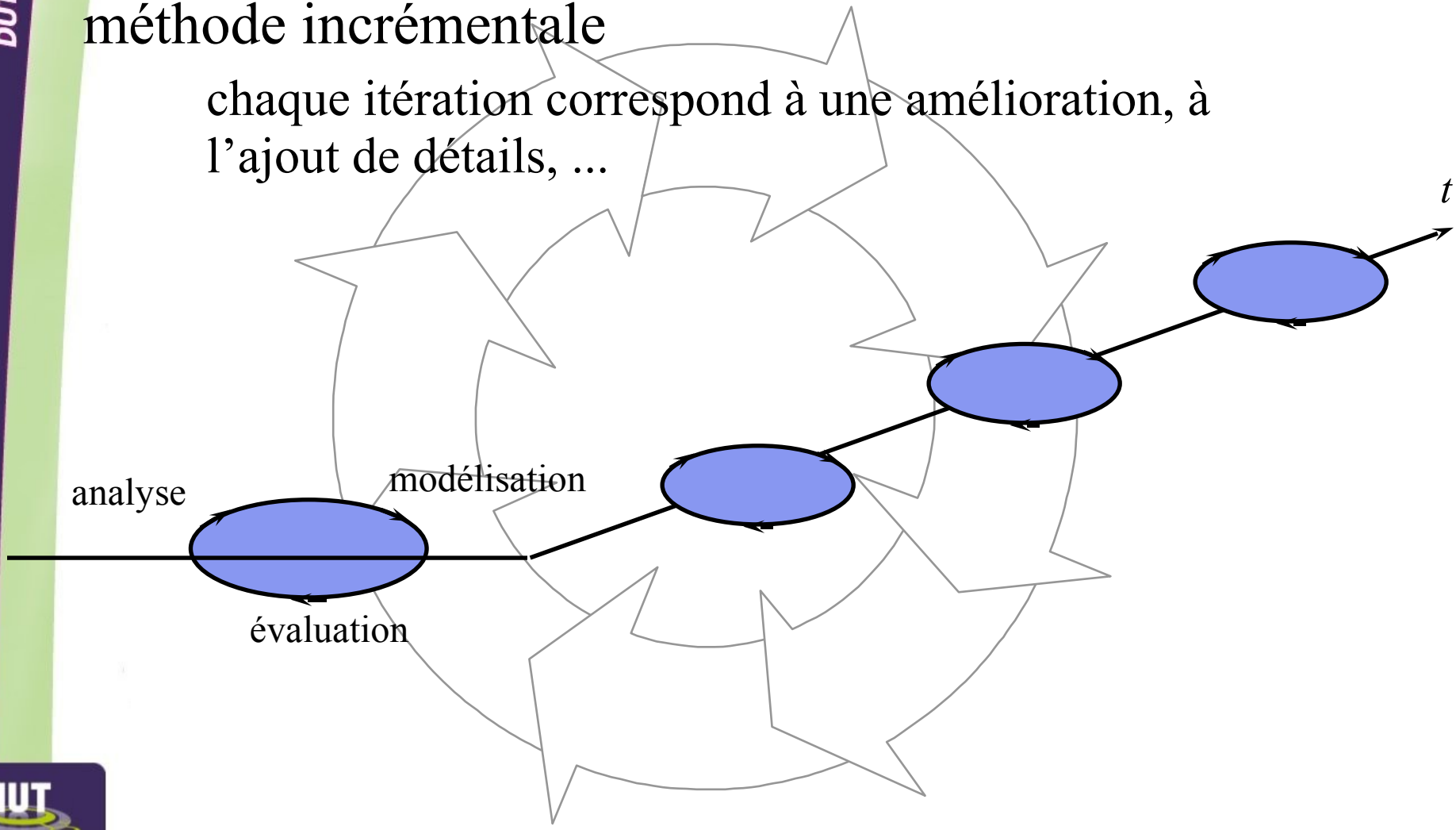
- guidé par les Use Cases
- centré sur l'architecture (la structure) du système
- itératif et incrémental (construction progressive)



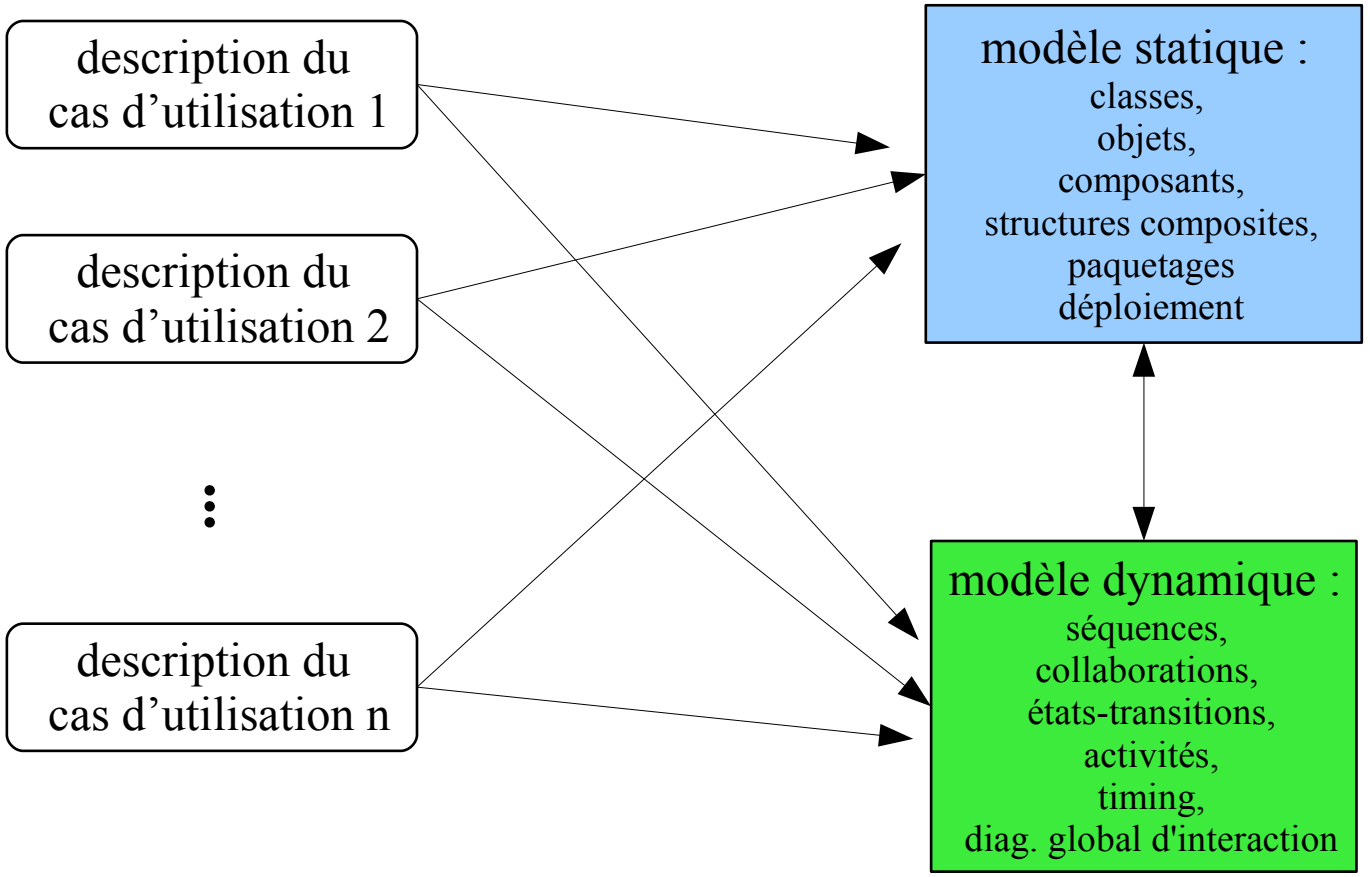
Méthode préconisée

méthode incrémentale

chaque itération correspond à une amélioration, à l'ajout de détails, ...

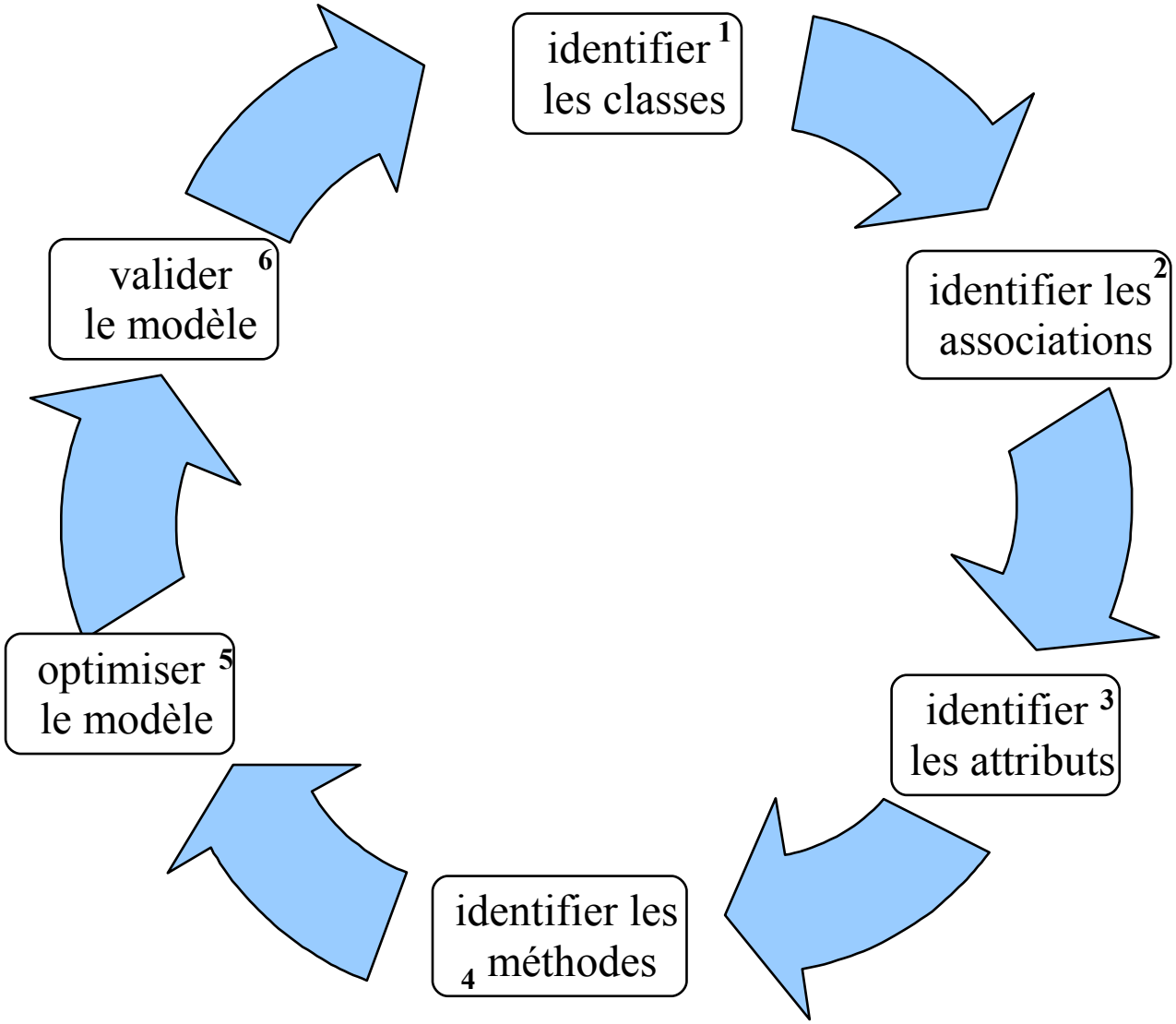


Synthèse de la méthode de modélisation

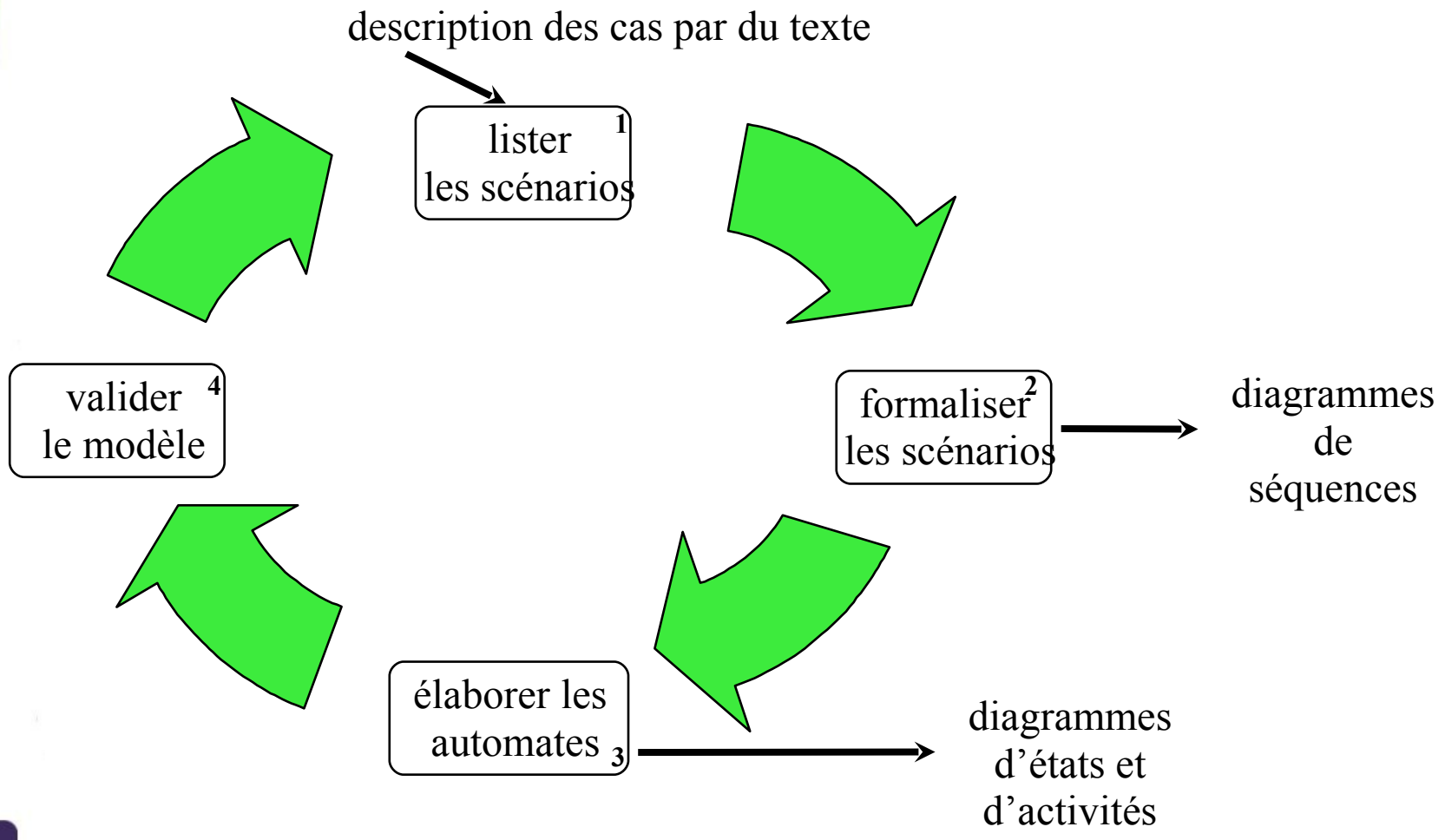


Tous les modèles doivent être **COHERENTS**

Construction du modèle statique :
ex. du diagramme de classes


identifier les classes 1
identifier les associations 2
identifier les attributs 3
identifier les méthodes 4
optimiser le modèle 5
valider le modèle 6

Construction du modèle dynamique


description des cas par du texte
lister les scénarios 1
formaliser les scénarios 2
diagrammes de séquences
élaborer les automates 3
diagrammes d'états et d'activités
valider le modèle 4

PLUS DE 40 ans D'EXPERIENCE
depuis 1967
iut
VALENCIENNES
CAMBRAI - MAUBEUGE
Informatique
DUT INFO

Cas d'utilisation

Analyse des besoins

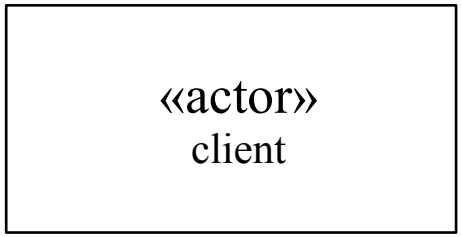
- Première analyse des besoins :
  - acteurs (types d'utilisateurs)
    - les acteurs sont externes au système.
      - =>Déterminer les limites du système
    - représentation des **acteurs**
  - cas d'utilisation (use cases)
    - pour chaque acteur :
      - Quels sont les services attendus ?
      - = Dans quel cas l'acteur utilise le système ?
      - Service = fonctionnalités Service ≠ action !!!!
    - représentation par :
      - cas sur le diagramme
      - liens acteurs-cas
      - du texte

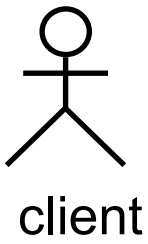
Acteur

Acteur = entité externe agissant sur le système :

être humain, machine, autre système ou sous-système

- L'acteur peut consulter et/ou modifier l'état du système
  (les entités externes passives ne sont pas des acteurs)
- un acteur est défini en fonction du **rôle** qu'il joue
  - ≠ (et non pas qui il est : toto)
- il représente un nombre quelconque d'entités externes agissant sur le système
- c'est un stéréotype de classe (classe «actor» prédéfinie)





Acteur - Exemple

Développement d'un système de caisse enregistreuse pour un supermarché.

Les acteurs possibles : client, caissier, gérant, système de gestion des stocks.

- l'acteur peut consulter et/ou modifier l'état du système
  (ex. : un vendeur du supermarché n'est pas un acteur pour ce système)
- il représente un nombre quelconque d'entités externes agissant sur le système : plusieurs utilisateurs peuvent correspondre à un seul acteur
  (ex. : les caissiers du supermarché représentés par un seul acteur)
- un acteur est défini en fonction du rôle qu'il joue : un utilisateur peut correspondre à plusieurs acteurs s'il joue différents rôles vis-à-vis du système
  (ex. : le gérant peut aussi être caissier)
- un acteur n'est pas nécessairement un être humain
  (ex. : le système de gestion des stocks)
- possibilité de classer les acteurs
  (ex. : client peut être soit un client individuel soit une société)

PLUS DE 40 ans D'EXPERIENCE depuis 1967
iut VALENCIENNES CAMBRAI • MAUBEUGE
Informatique DUT INFO

Acteur / Personne ?

les personnes sont souvent représentées par 3 éléments du modèle :

- des acteurs : utilisateurs du système
- des objets (objets métier) : informations décrivant chaque utilisateur
- des objets d'interfaces : manipulation des informations contenues par les objets métier

IUT

Cas d'utilisation

- décrivent les fonctionnalités fournies par le système à un acteur.
- modèle du système du point-de-vue de son utilisation, de ses fonctionnalités :

Que devrait faire le système ?

  - Interactions entre le système et les acteurs
  - Réactions du système aux événements extérieurs

Les cas d'utilisation définissent :

- les limites du système (interne au système/ externe)
- le comportement attendu du système

Cas d'utilisation

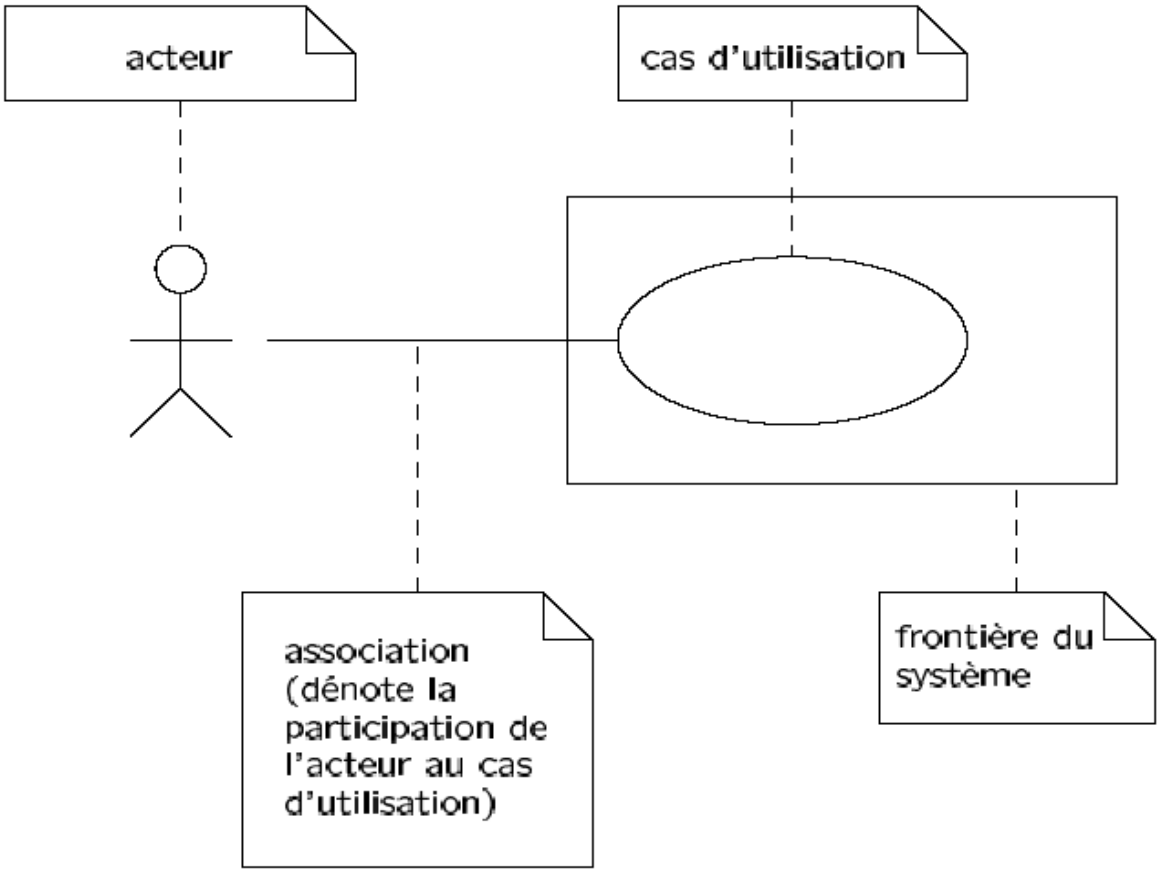


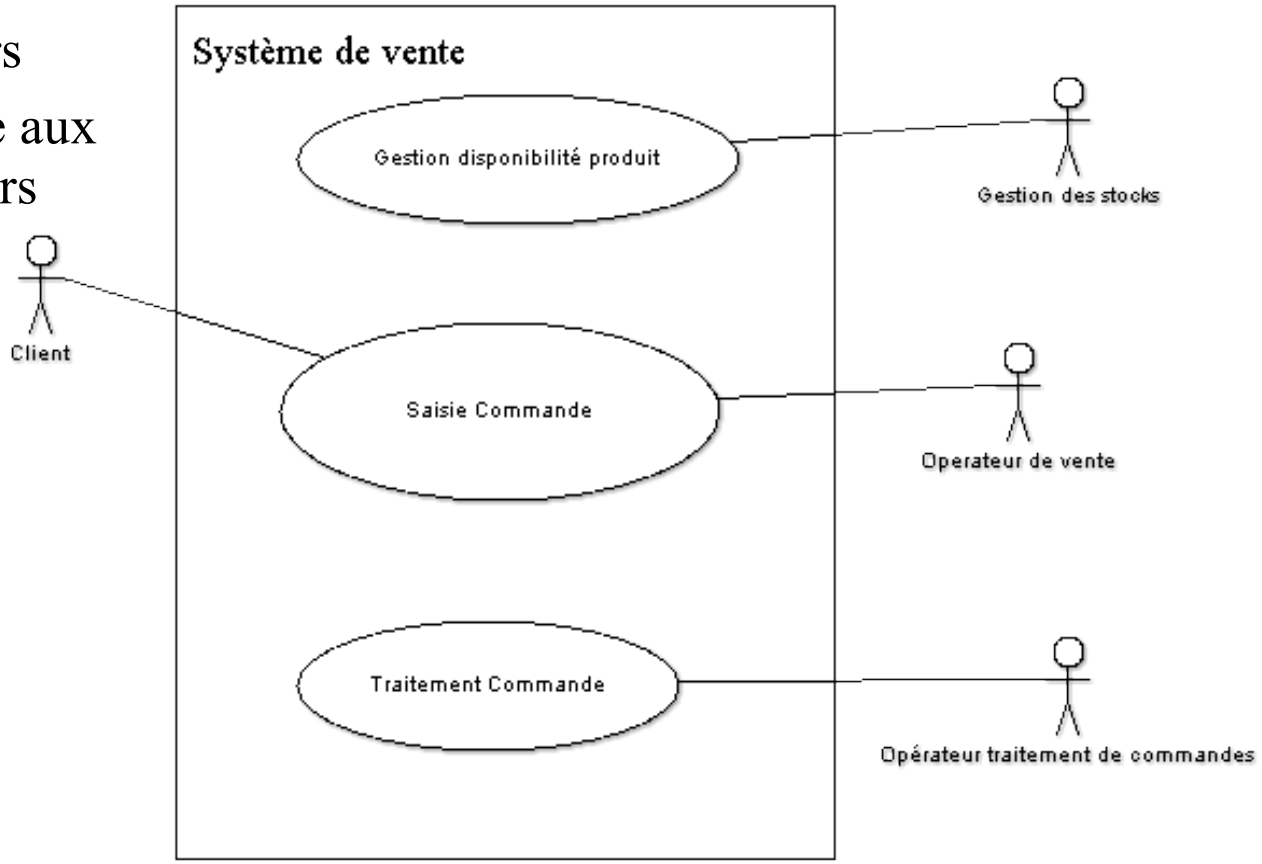
Diagramme des Cas d'Utilisation

(Use Cases)

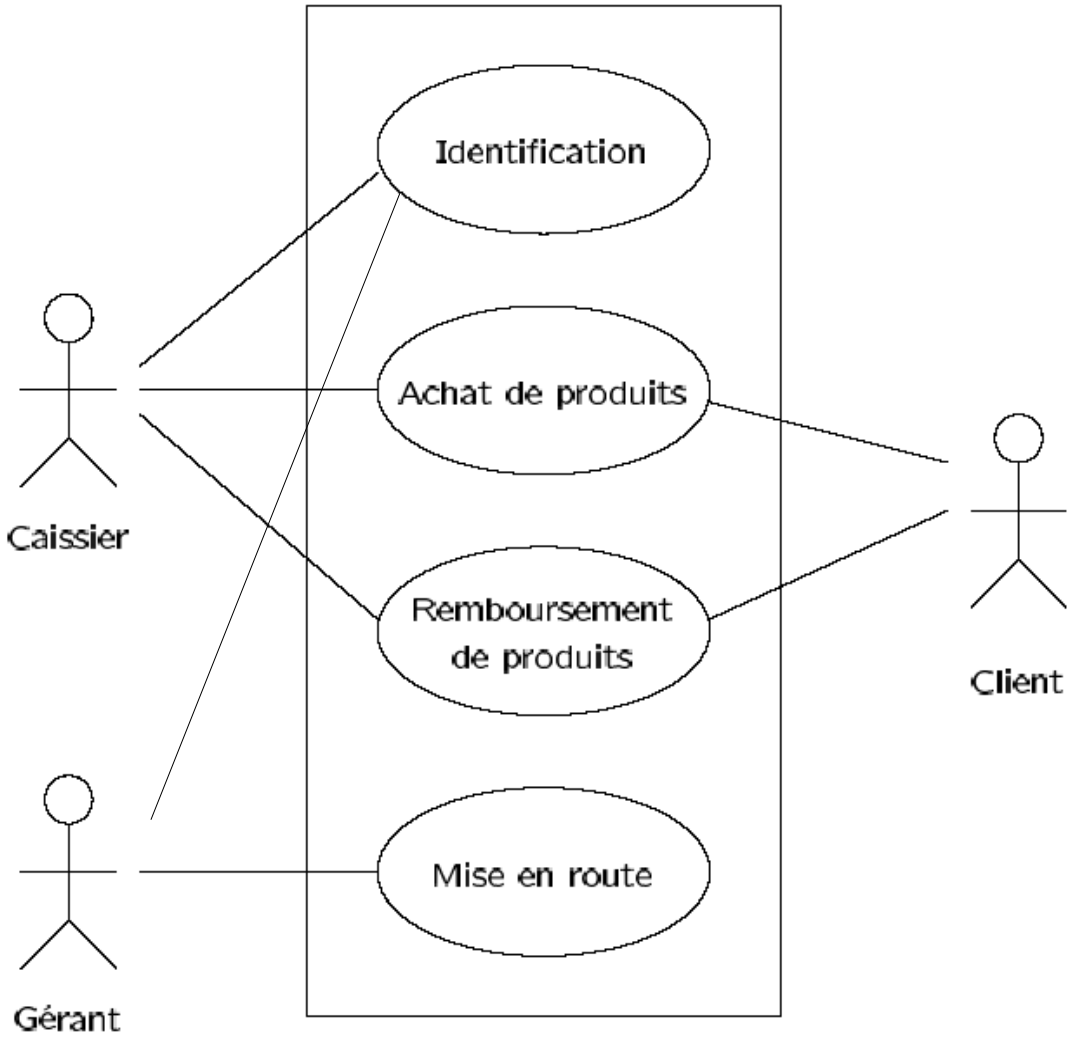
Modèle du système du point de vue de son utilisation (fonctionnalités) :

– interactions entre le système et les acteurs

– réactions du système aux événements extérieurs



Cas d'utilisation – Exemple d'une caisse enregistreuse



Cas d'utilisation – Description textuelle

- Le diagramme de cas d'utilisation est accompagné d'une description :

  Chaque cas d'utilisation est décrit par du texte :

Nom	Nom du cas d'utilisation
Acteurs	Liste des acteurs qui interviennent pour ce cas
But	Objectif du cas d'utilisation (que fait-il)
Pré-conditions	Les conditions nécessaires pour déclencher le cas
Post-conditions	Les conditions de sortie : état du système après réalisation du cas

Cas d'utilisation – Description textuelle

Et par des scénarios :

N°	Action des acteurs	Actions du système
1	Action a.1	Action b.1
2	Action a.2	Action b.2
...	...	...

3 types de scénarios possibles :

- **Scénario nominal** : scénario idéal ou « normal »
- **Scénario alternatif** : une variante d'un scénario nominal (une autre façon de faire).
  Les post-conditions sont remplies.
- **Scénario d'exception** : une variante qui conduit à ne pas remplir les post-conditions du cas

Ex. d'une caisse enregistreuse

Nom	Identification
Acteurs	Caissier, Gérant
But	Sécuriser l'utilisation de la caisse en : - Identifiant la personne qui va s'en servir - Contrôlant le dévérouillage de la caisse
Pré-conditions	Les identifications des caissiers sont enregistrées
Post-conditions	Le caissier est identifié et la caisse dévérouillée

Ex. d'une caisse enregistreuse

Scénario nominal

N°	Action des acteurs	Actions du système
1		Le système demande au caissier de s'identifier
2	Le caissier s'identifie	
3		Le système vérifie l'identification
4		Le système dévérouille la caisse

Ex. d'une caisse enregistreuse

Scénario alternatif

N°	Action des acteurs	Actions du système
3.1		L'identification est inconnue et la demande a été faite moins de 3 fois : - Le système redemande l'identification - Reprise au point 2 du scénario nominal

Ex. d'une caisse enregistreuse

Scénario d'exception

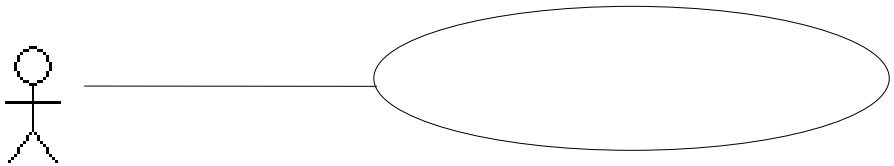
N°	Action des acteurs	Actions du système
3.2		L'identification est inconnue et la demande a été faite déjà 3 fois : - Le système bloque la caisse et l'indique au caissier - Le système alerte le gérant

Liens Acteur - Cas d'utilisation

- Lien de communication entre acteur et cas d'utilisation :

  Un acteur Participe A un cas d'utilisation

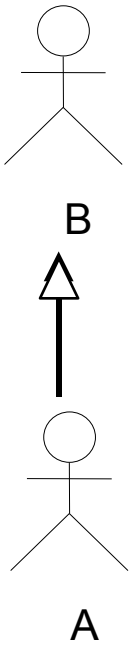
Seul lien possible entre acteur et cas d'utilisation.



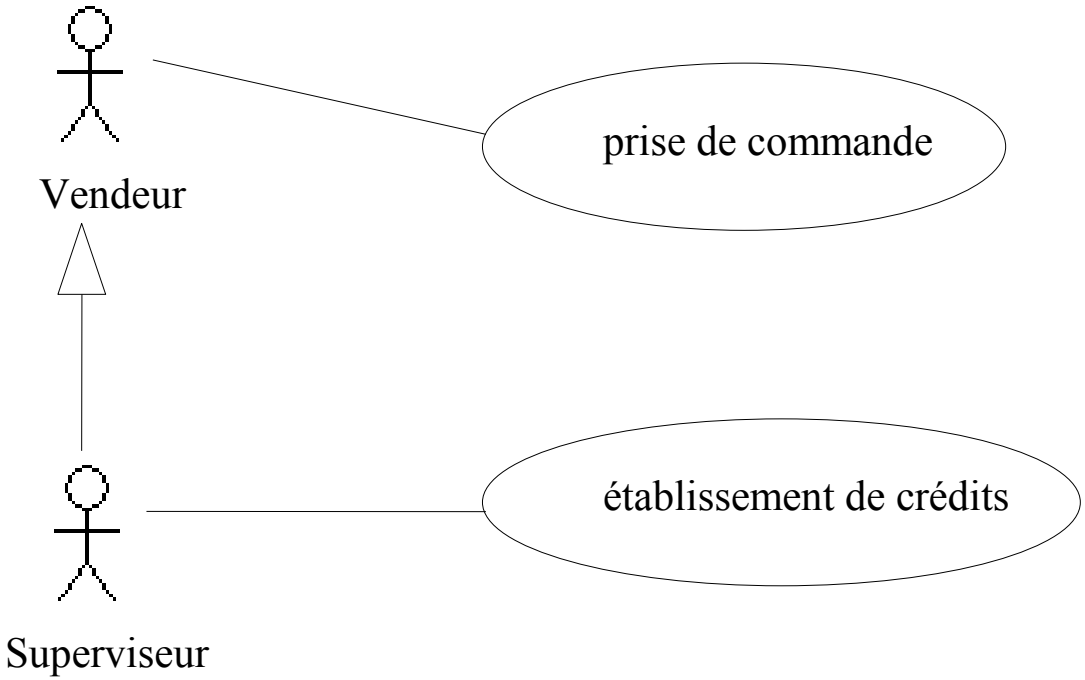
Liens entre Acteurs

- Lien de Généralisation de A vers B :

  A peut communiquer avec les mêmes cas d'utilisation que B



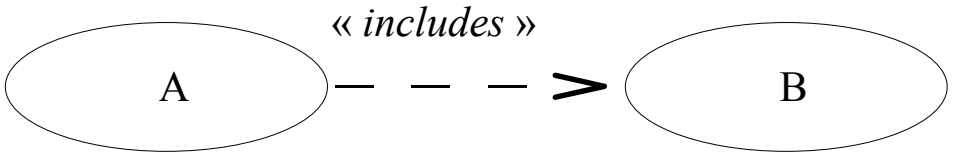
Exemple


prise de commande
Vendeur
établissement de crédits
Superviseur

Liens entre 2 cas d'utilisation

- 3 types de liens :
  - Relation d'inclusion (« includes »)
  - Relation d'extension (« extends »)
  - Relation de généralisation (« generalizes »)

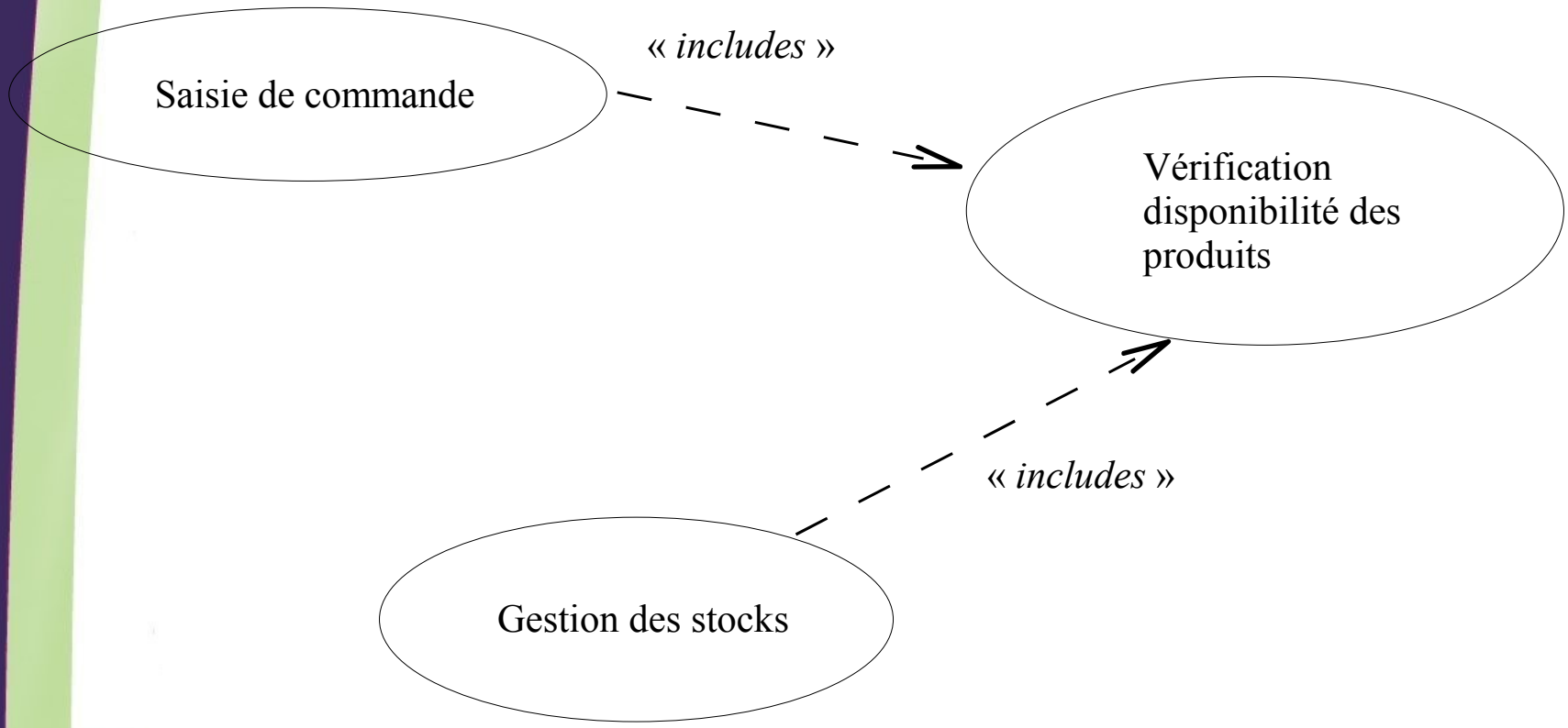
Relation d'inclusion



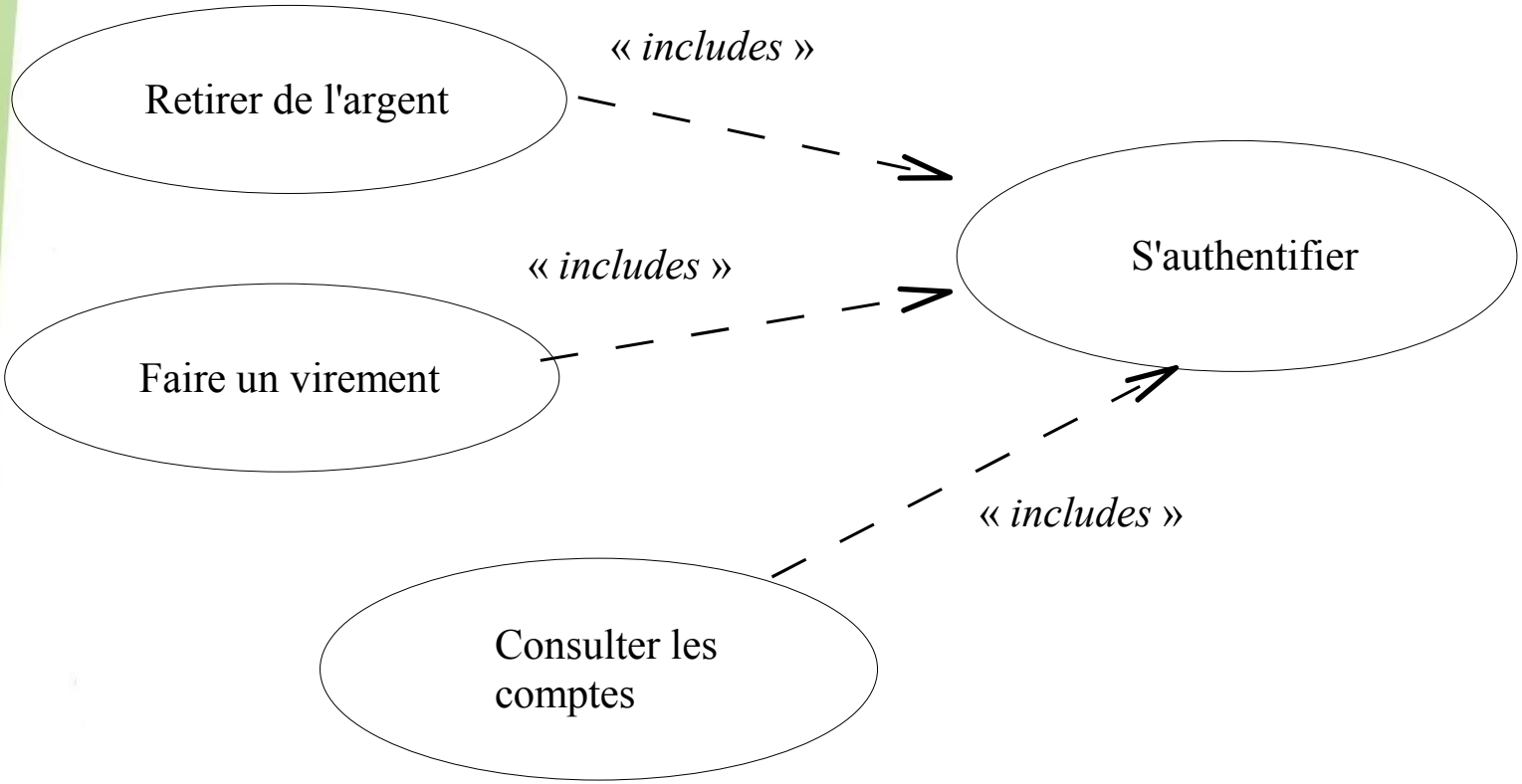
A inclut B
B est inclus dans A

- notée « include » ou « includes »
- B est une partie de A
- B est indispensable pour réaliser A
- A « utilise » B
- B n'est pas complet et autonome
- sert à partager une fonctionnalité (par ex. B) entre plusieurs cas d'utilisation

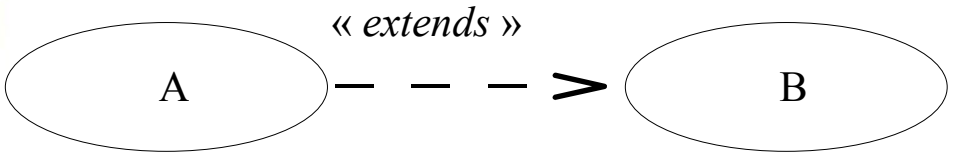
Exemple 1 – vente par correspondance



Exemple 2 – guichet automatique



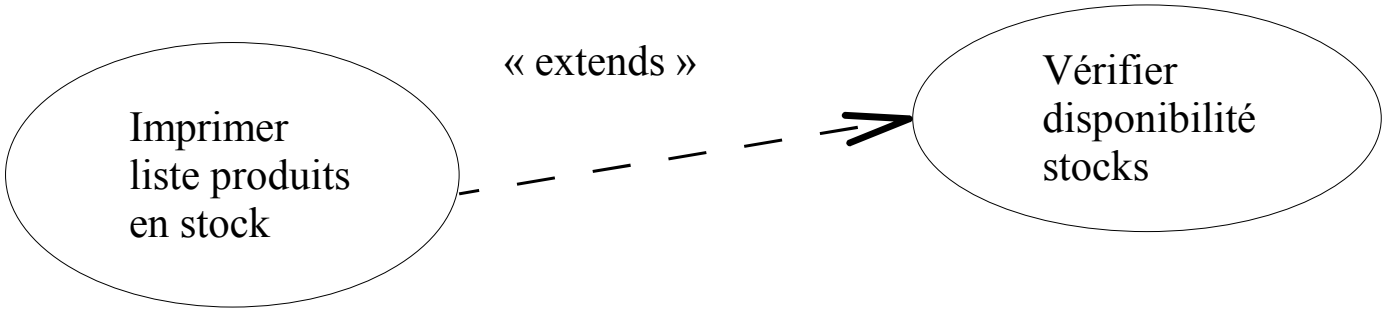
Relation d'extension



A étend B
A est une extension de B

- notée « extend » ou « extends »
- A comprend des actions supplémentaires à B
- B peut être étendu (augmenté) par le comportement spécifié par A.
- B peut être employé sans A (B est autonome)

Exemple 1

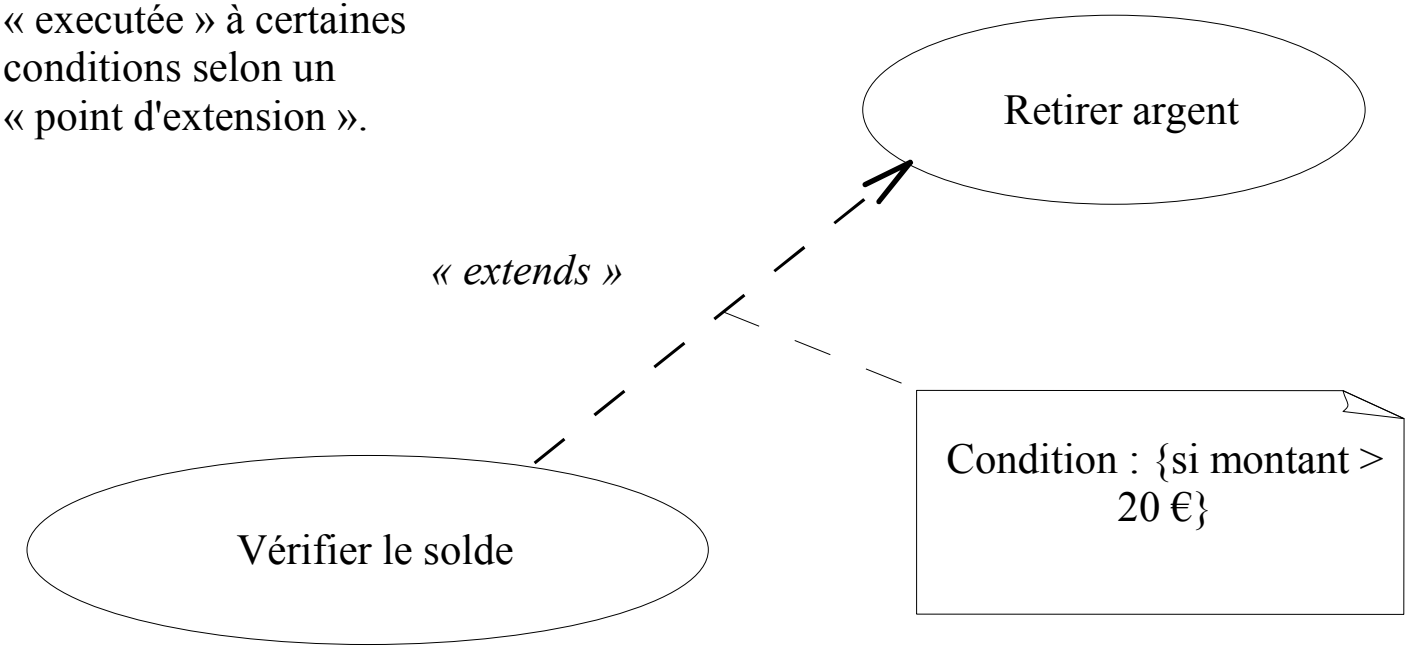


- Pourquoi « Extends » ?
  - « Imprimer... » comprend des actions supplémentaires à « Vérifier »
  - « Vérifier » peut être étendu (augmenté) par le comportement spécifié par « Imprimer ».
  - « Vérifier » peut être employé sans « Imprimer »

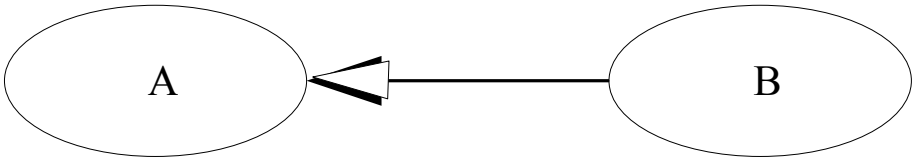
- Pourquoi pas « Includes » ?
  - «Vérifier... » n'est pas une partie de « Imprimer ... »
  - « Vérifier... » est indispensable pour réaliser « Imprimer ... ».
  - «Imprimer...» dépend de «Vérifier...»
  - « Vérifier... » est autonome

Exemple 2 – guichet automatique

L'extension peut être « executée » à certaines conditions selon un « point d'extension ».



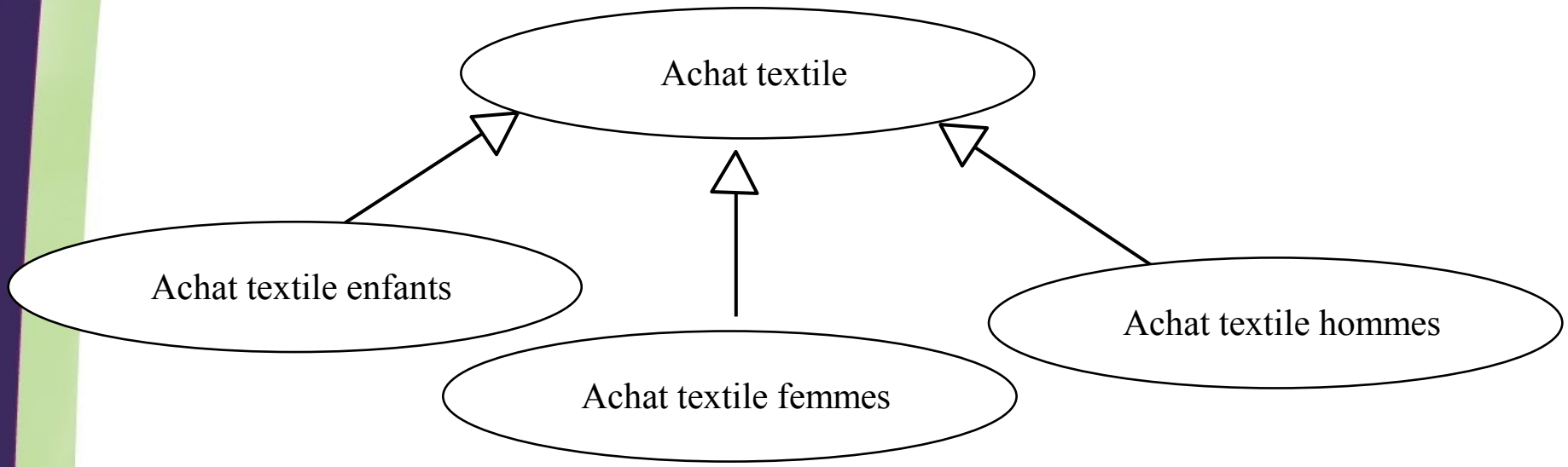
Relation de généralisation



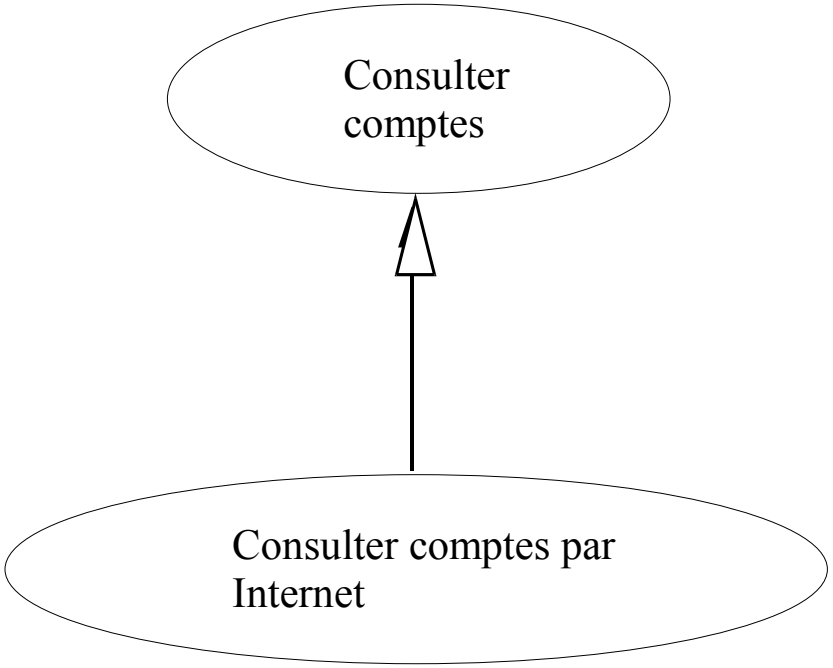
A généralise B
B est une spécialisation de A

- notée « generalizes »
  (notation optionnelle)
- A généralise B signifie que A est plus abstrait que B
- B est une spécialisation de A

Exemple 1


Achat textile
Achat textile enfants
Achat textile femmes
Achat textile hommes

Exemple 2 – banque



Modélisation Statique

ou

diagrammes structurels

classes,
objets,
composants,

structures composites,

paquetages

déploiement

Modélisation statique

- Objectif : faire un modèle montrant les éléments qui vont composer le système et leur organisation (les liens entre eux).
- Diagrammes :
  - D'objets
  - de classes
  - de composants
  - de structures composites
  - de paquetages
  - de déploiement

iut VALENCIENNES CAMBRAI - MAUBEUGE
depuis 1967
40 ans d'expérience
Informatique DUT INFO

Objet

- objet = représentation d'une entité physique ou abstraite
- Un objet est formé d'un état et d'un comportement
  - un **état** : l'ensemble des valeurs de ses caractéristiques à un instant donné

    représenté par des **attributs**
  - un **comportement** : manière dont l'objet agit et réagit.

    représenté par des **opérations** ou **méthodes**

Ex. d'objets informatiques

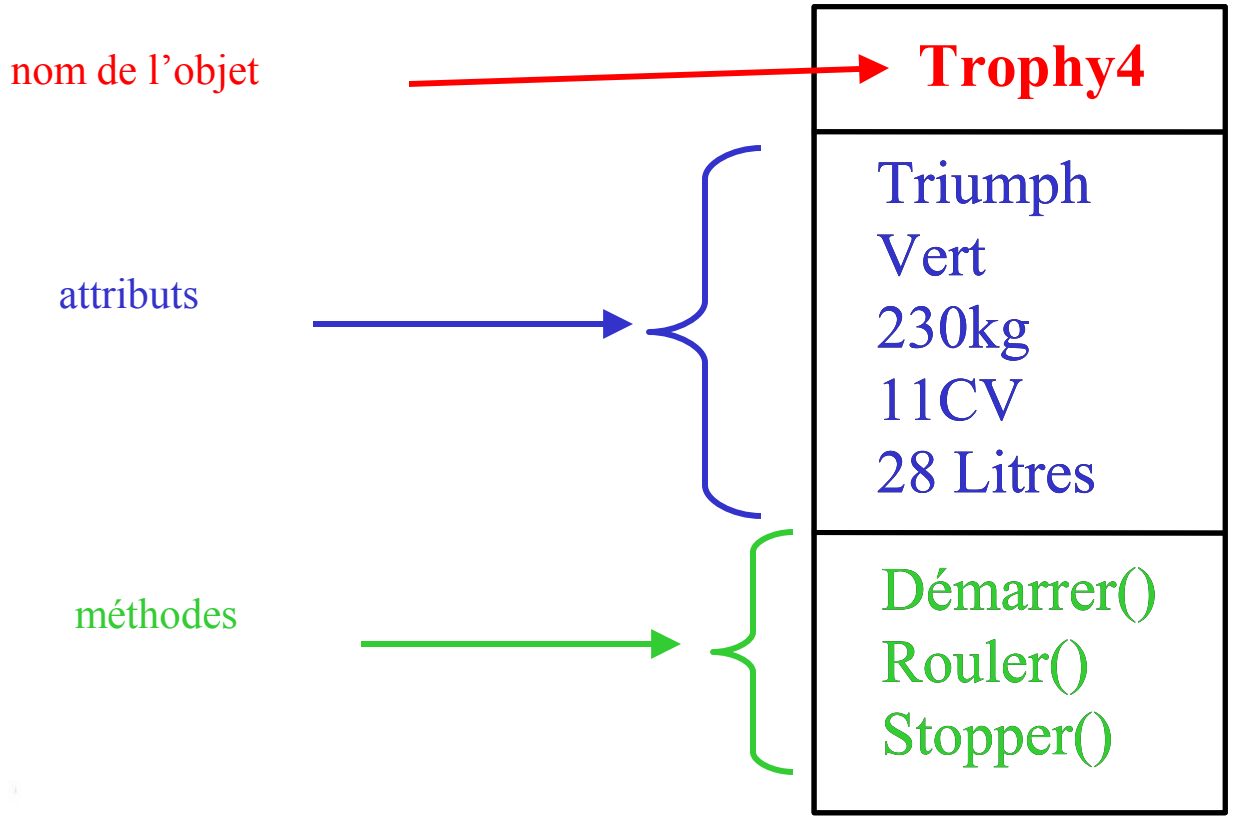
- objets matériels :

  porte, ascenseur, bouton, clavier, souris, avion, ...
- différents objets, dans une entreprise :

  compte en banque, équation mathématique,
  accueil, commercial, contrôleur qualité,
  groupe d'abonnés à un forum internet,
  facture, commande, marché, ...
- pour un logiciel de gestion de la clientèle :

  client, requête SQL,
  bouton, fenêtre, ...

Représentation d'un objet



Diagrammes d'objets

- Objectif :
  - Obtenir des instantanés montrant l'état des transactions interne au système pendant son execution
- Contenu :
  - Objets + liens entre les objets à un moment précis de l'execution

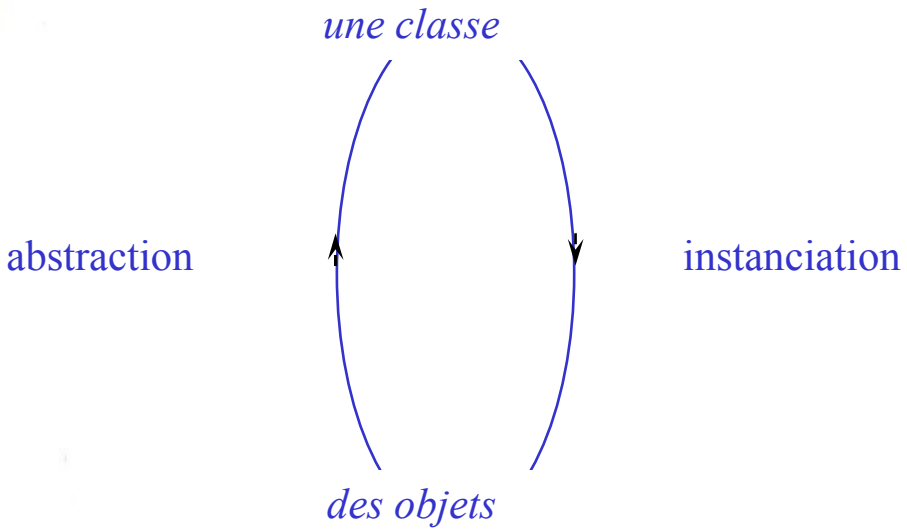
Exemple

Employe 1
Dupont Albert Toulouse 2000 euros
Modifier_Adr() Modifier_Salaire()

Employe 2
Martin André Bordeaux 2300 euros
Modifier_Adr() Modifier_Salaire()

Classe

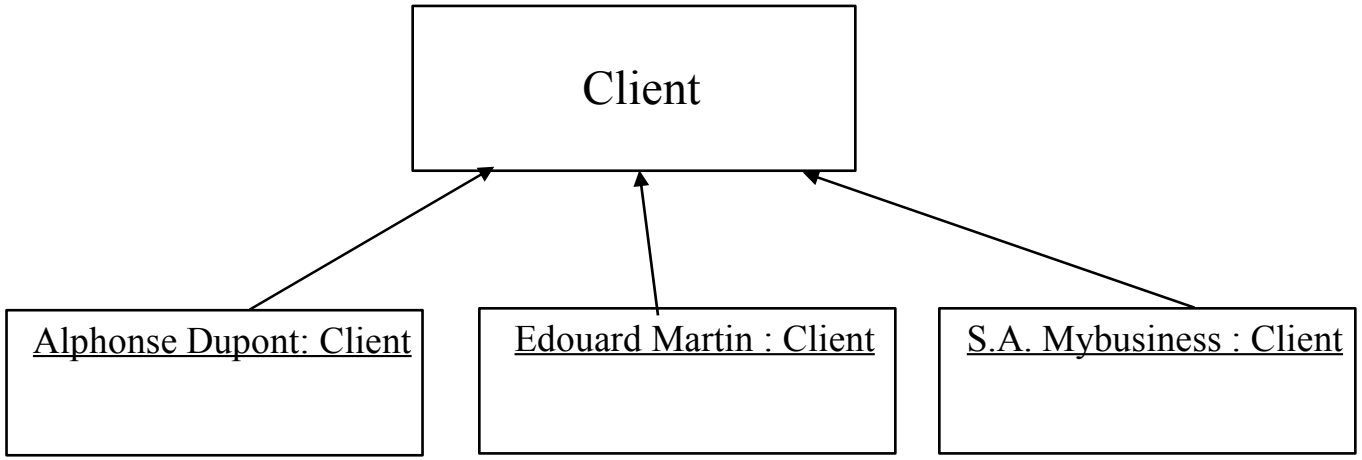
- classe regroupe des objets qui se ressemblent
- modèle pour créer plusieurs objets présentant des caractéristiques communes (les instances)



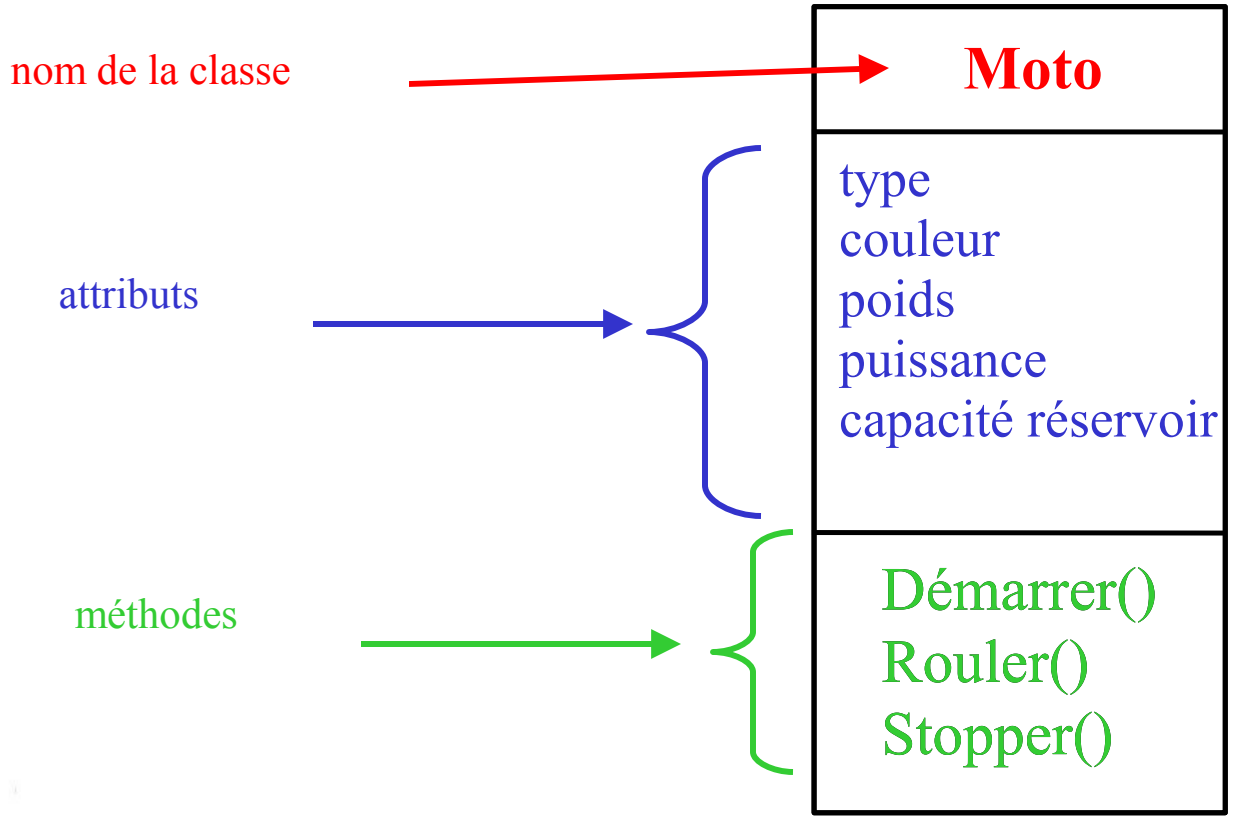
un objet est une instance d'une classe

Ex.

- *classes* : rôles des salariés
  instances : individus ayant ces rôles
- *classes* : facture, commande
  instances : facture X, facture Y, commande XX, ...
- *classe* : client de la banque
  instances : A. Dupont, E. Martin et S.A. Mybusiness



Représentation d'une classe



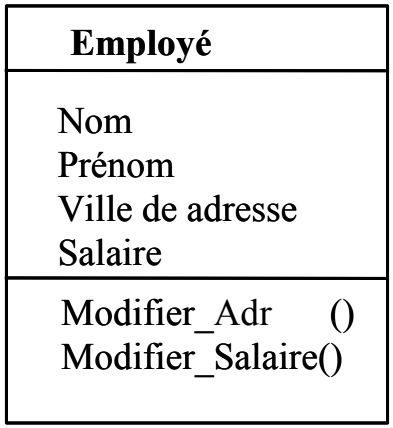
Classes

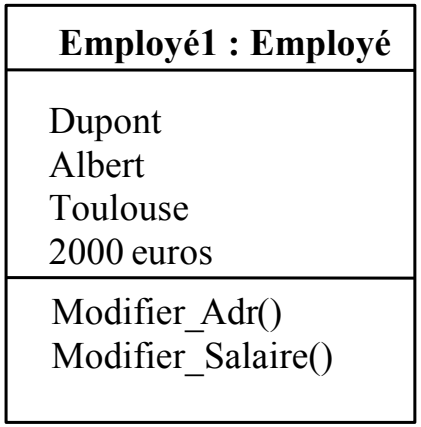
Point
x,y : Integer

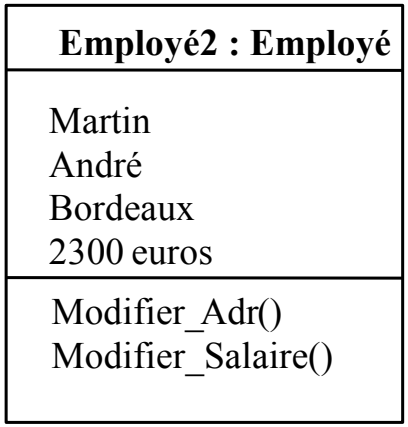
Segment
origine, extremite : Point
longueur() : Integer deplace(dx,dy : Integer)

Employe
Nom Prenom VilleDeAdresse Salaire
Modifier_Adr() Modifier_Salaire()

Ex. classe / objets instances

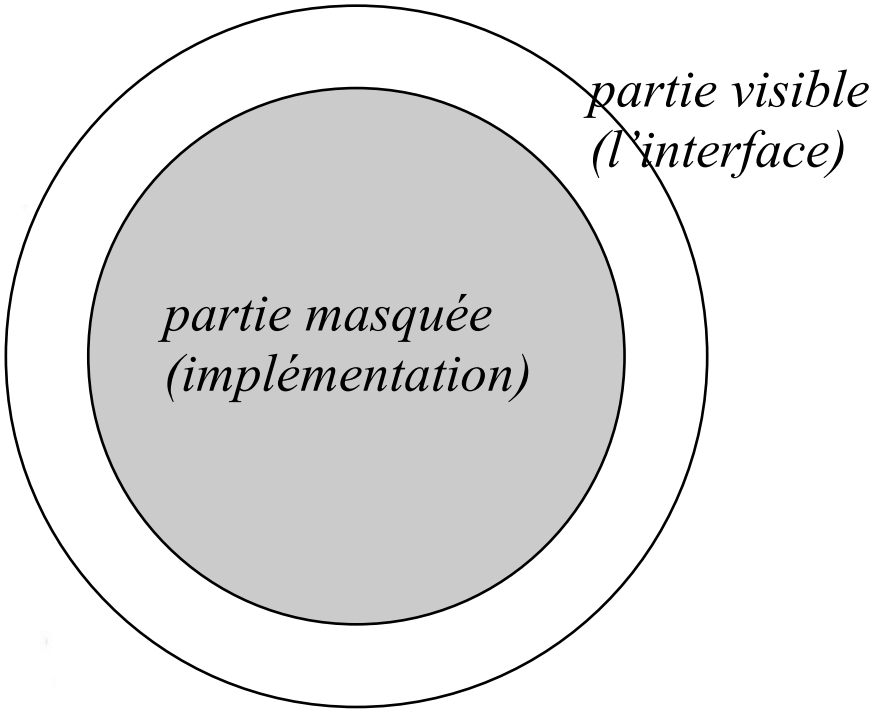






Encapsulation

principe : masquer la réalisation.



Objectifs :

- garantit l'intégrité et la sécurité des données
- facile à maintenir (changements dans l'implémentation sans modifier l'interface)

Encapsulation

L'encapsulation permet de protéger des attributs par rapport à des accès non autorisés : certains attributs peuvent être cachés

Ex.

On ne veut pas que l'adresse dans la classe *Employée* puisse être lue ou modifiée directement.

La seule façon de faire autorisée est de passer par la méthode *modifier_adr()*.

Cela permet de contrôler les valeurs données.

Encapsulation

L'encapsulation permet de changer des attributs, des méthodes ou le code des méthodes sans que l'utilisateur de la classe n'ait à modifier son code.

Ex.

Supposons que l'adresse dans la classe *Employée* ci-dessus soit codée en tant que chaîne de 20 caractères. L'accès à la classe employée via *modifier_adr()* n'est pas modifiée même si entre temps le type de l'attribut *Adresse* a été modifié (par ex., passage à 2 chaînes de 10 caractères).

Protection et droit d'accès

- Les attributs et les méthodes membres d'une classe peuvent être protégées par des droits d'accès.
- 3 types de droits :

  - private : accessibles uniquement aux objets de la classe

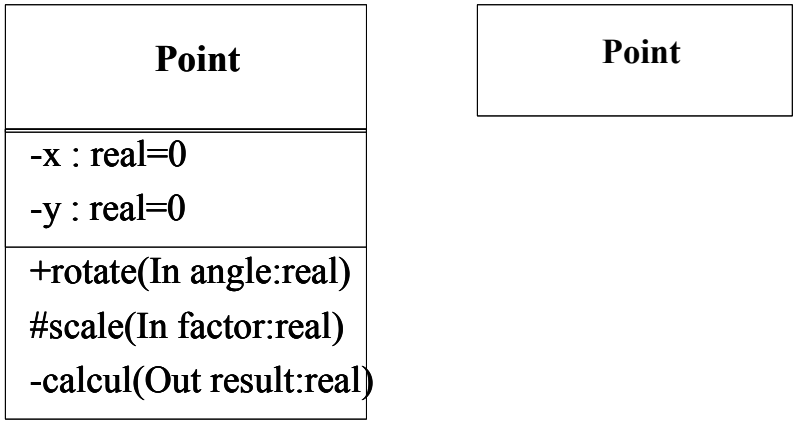
  # protected : accessibles uniquement aux objets de la classe et aux objets des sous-classes (selon le type d'héritage)

  + public : accessibles par tous.

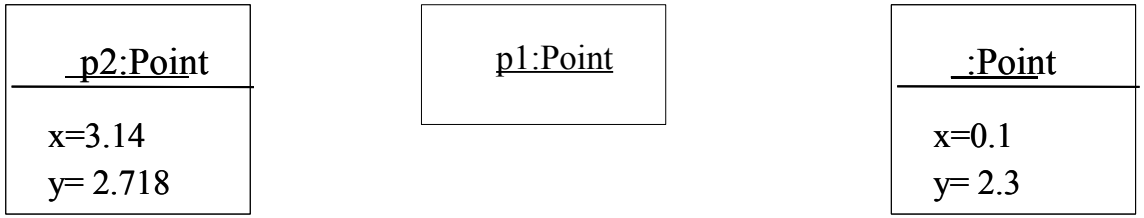
Ex représentations de classes et d'objets

Différentes représentations plus ou moins détaillées :

classes :



objets :

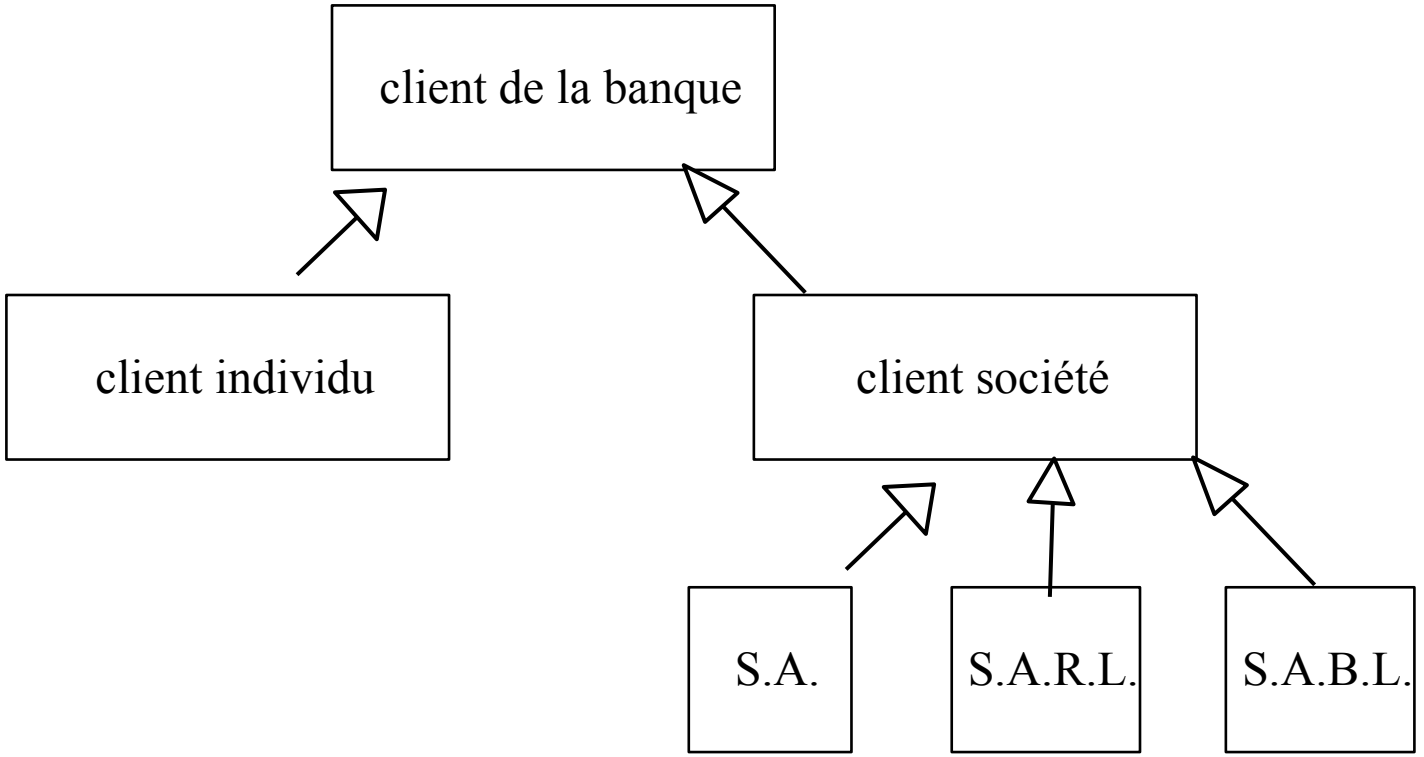


iut VALENCIENNES CAMBRAI - MAUBEUGE
depuis 1967
+40 ans d'expérience
Informatique DUT INFO

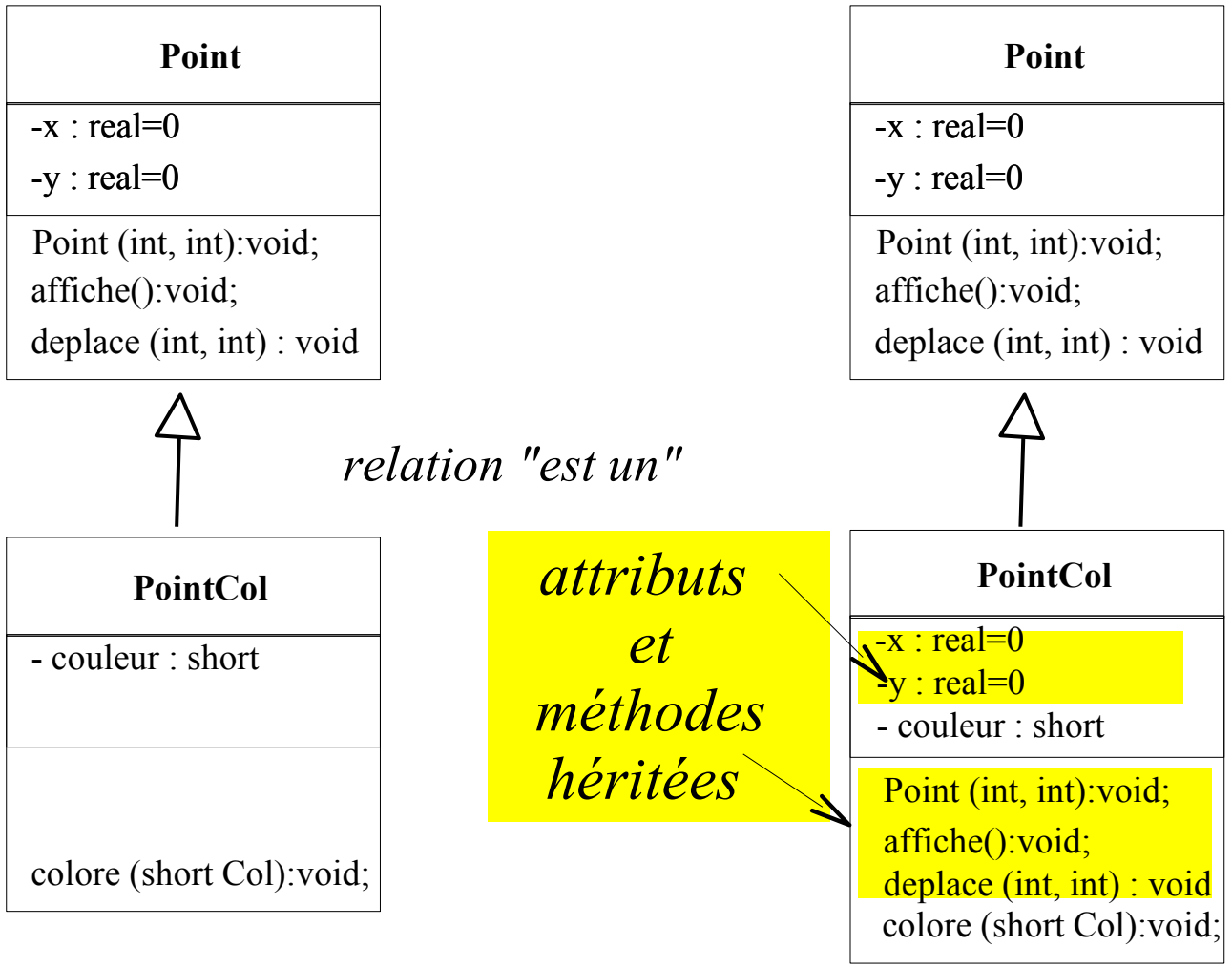
Héritage

- **héritage** : transmission des caractéristiques d'une classe vers ses sous-classes
- classement des abstractions en **hiérarchie**
- permet de définir de nouvelles classes à partir d'une classe de base existante :
  ajout de nouvelles données et de nouvelles méthodes.
- La ou les classes dérivées "spécialisent" la classe de base.

Ex. de hiérarchie de classe



Ex. Héritage


Point
-x : real=0
-y : real=0
Point (int, int):void;
affiche():void;
deplace (int, int) : void
Point
-x : real=0
-y : real=0
Point (int, int):void;
affiche():void;
deplace (int, int) : void
relation "est un"
PointCol
- couleur : short
colore (short Col):void;
attributs et méthodes héritées
PointCol
-x : real=0
-y : real=0
- couleur : short
Point (int, int):void;
affiche():void;
deplace (int, int) : void
colore (short Col):void;

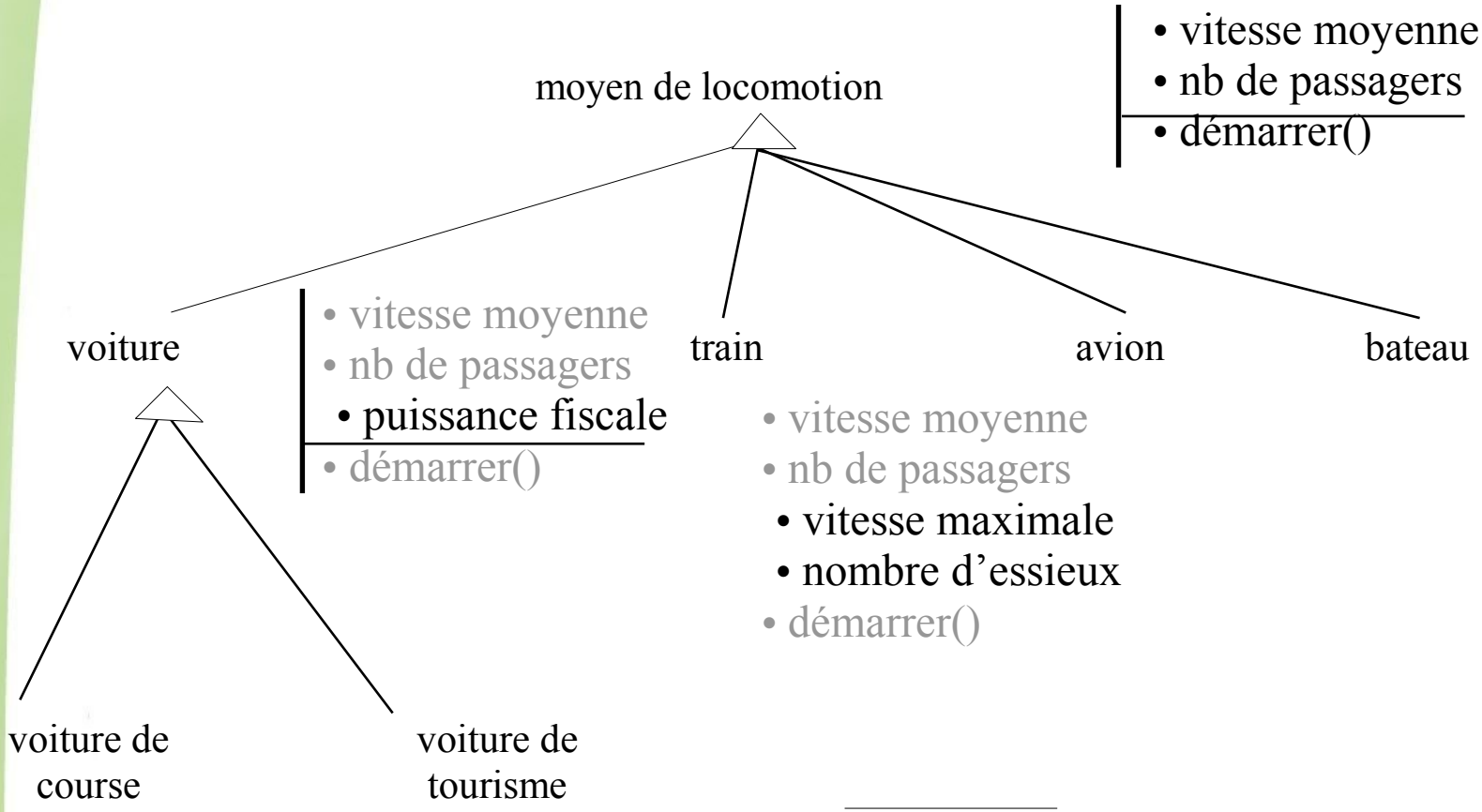
Exercice

- Faire une hiérarchie avec les classes suivantes :

  voiture, train, moyen de locomotion, avion, voiture de course, bateau, voiture de tourisme.
- Donner qqs attributs et méthodes.
- Ajouter voiture amphibie, radar et Awacs à la hiérarchie précédente.

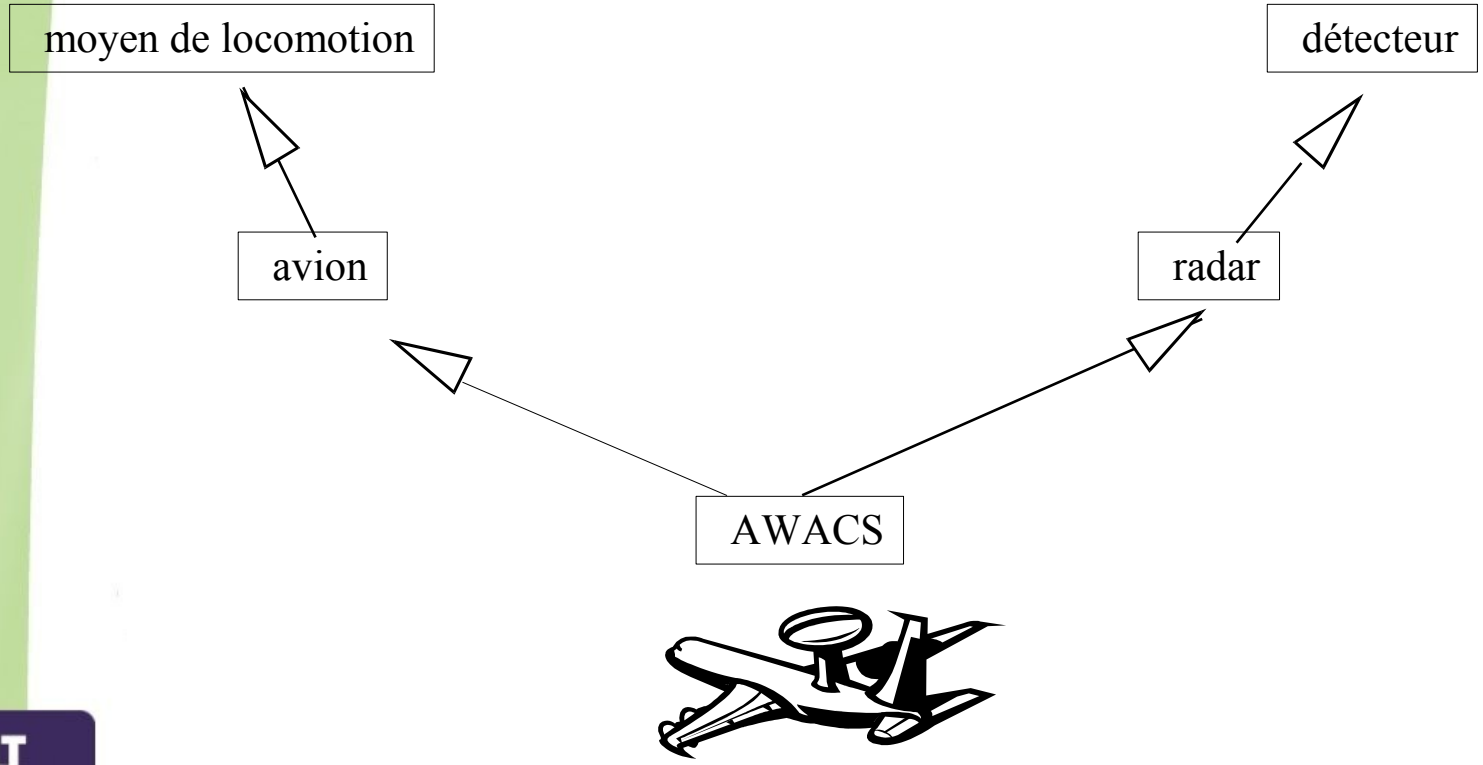
Ex.

- *héritage simple : «est un»*



Ex.

- *héritage multiple : «partie de»*



Association entre classes

- *Université* « connaît » *Personne*
- *Personne* « connaît » *Université*

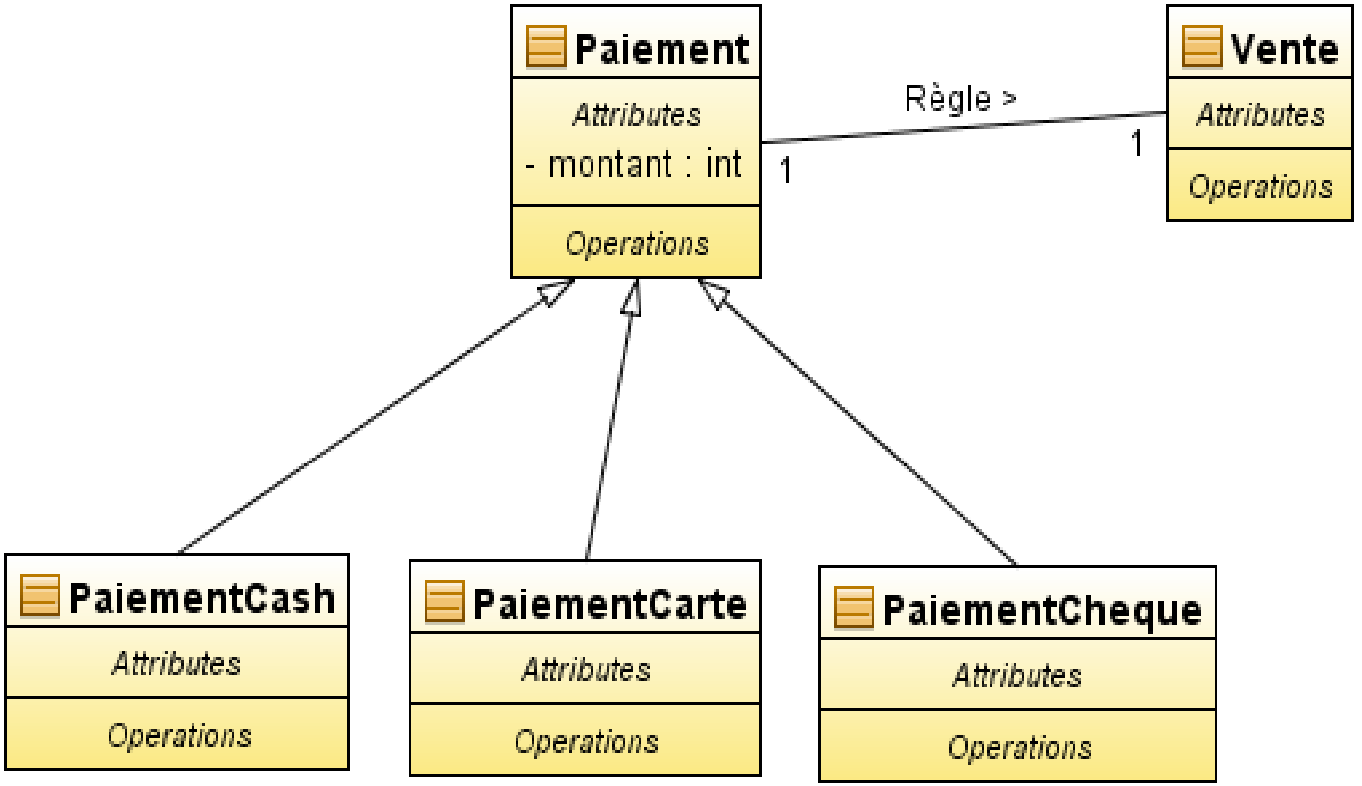
Association entre classes

- Navigabilité restreinte représentée par une flèche :
  - *Université* « connaît » *Personne*
    mais l'inverse est faux

Cardinalités

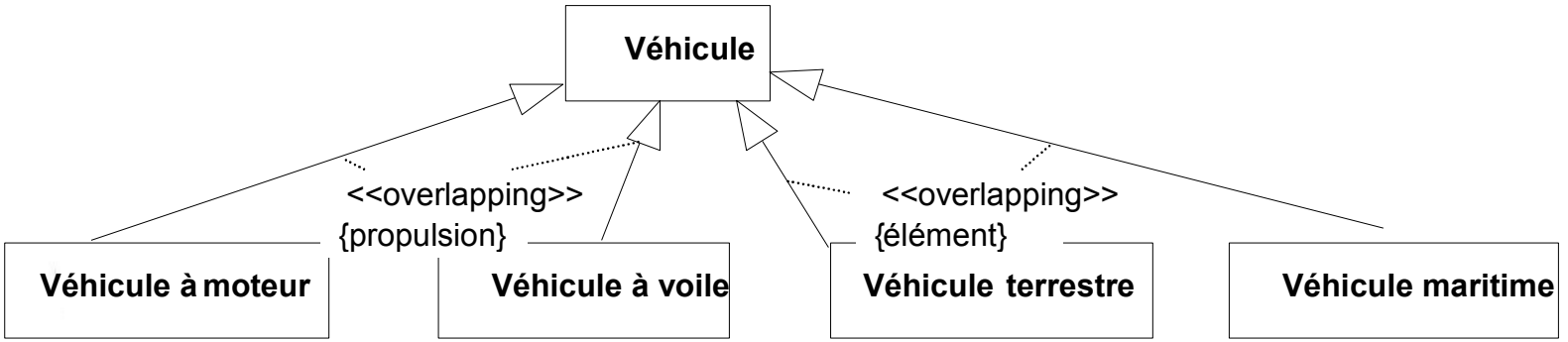
- Cardinalités et rôles peuvent être définis (c'est facultatif) :
  - **0..1** 0 ou 1
  - ***** plusieurs non défini (infinité)
  - **1** 1 et 1 seul
  - **4** n nombre précis défini
  - **3..*** n à infinité
  - **1..3, 7..10, 15, 19..*** combinaison d'intervalles et de valeurs précises

Ex. avec héritage et association



Discriminateurs de classes héritées

- **Discriminateur** : critère de distinction entre 2 classes spécialisées :
  - overlapping (intersection non nulle entre sous-ensembles)
  - disjoint (intersection nulle)
  - complete (tous les sous-ensembles sont spécifiés)
  - incomplete (liste de sous-ens. est incomplète)



Relation d'agrégation

- Ex. : une entreprise peut contenir un ensemble de services

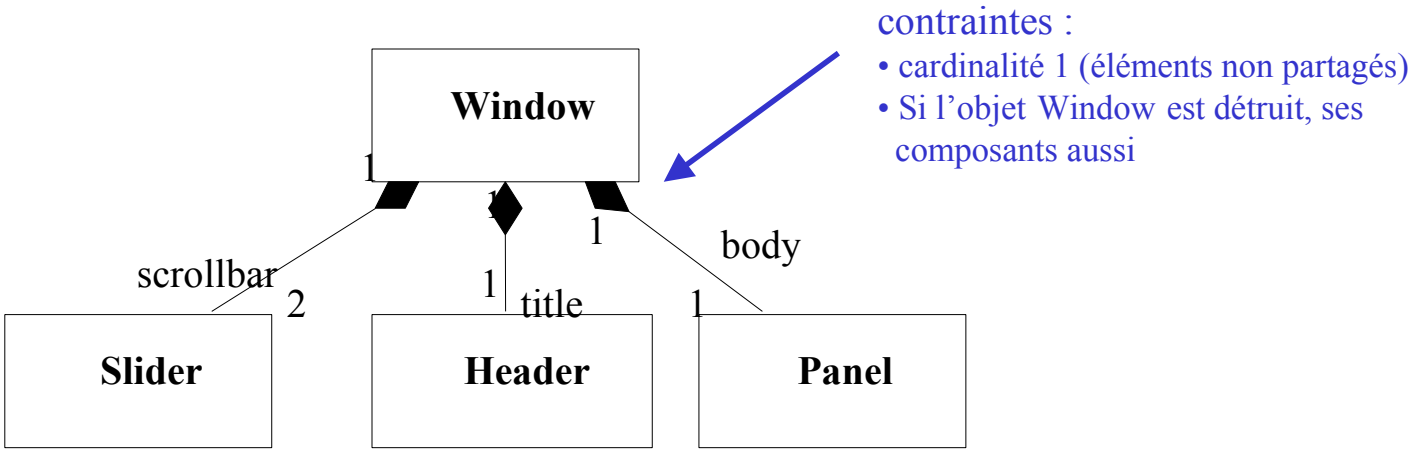
Rem. : Un composant peut faire partie de plusieurs agrégats

Dans l'exemple : Un service peut être utilidés dans plusieurs entreprises

(dans ce cas il faut mettre la cardinalité adaptée)

Composition

- Composition = agrégation forte
- Un composant peut faire partie d'un et un seul composite.
- Cette relation exprime souvent l'appartenance physique de parties à un tout.



Exemple diag. de classes

Ex. d'objets graphiques :

- Une forme graphique comporte un polygone, une couleur de remplissage et une couleur de trait.
- Un polygone comporte plusieurs segments.
- Une palette graphique comporte un ensemble de couleurs

Quelles questions faut-il se poser pour différencier agrégation et composition ?

Exemple diag. de classes

- <u>Ex. d'objets graphiques :</u>
- Une forme graphique comporte un polygone, une couleur de remplissage et une couleur de trait.
- Un polygone comporte plusieurs segments.
- Une palette graphique comporte un ensemble de couleurs

Les éléments sont-ils détruits lorsqu'on détruit l'ensemble ?

Le polygone est-il détruit en même temps que la forme graphique ?
Existe-t-il indépendamment de la forme graphique ?

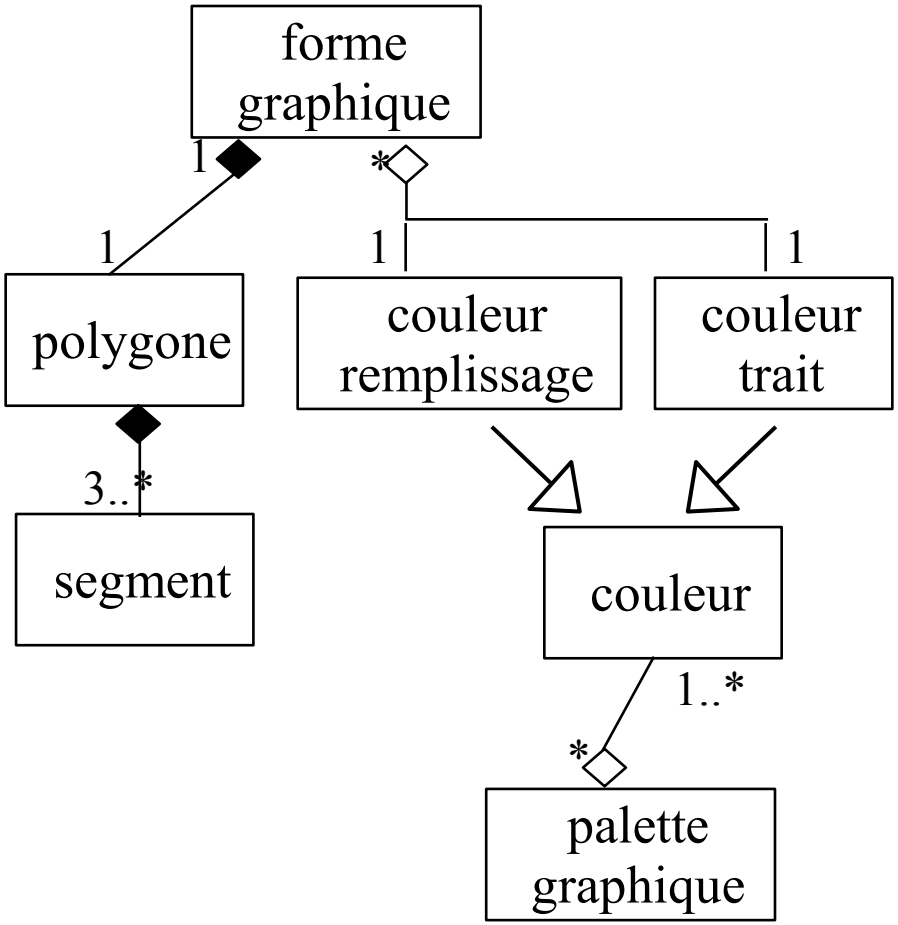
Un élément peut-il faire parti de plusieurs ensembles ?

Un polygone peut-il faire parti de plusieurs formes graphiques ?

Exemple diag. de classes

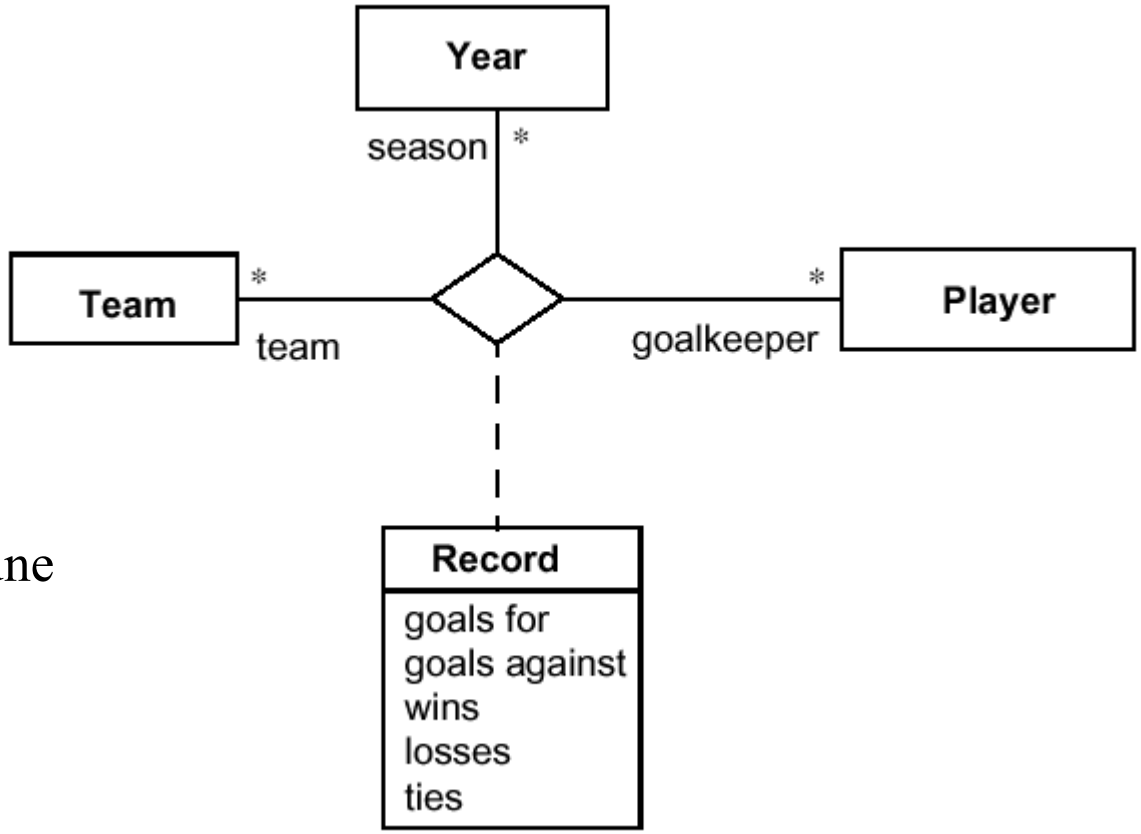
- <u>Ex. d'éléments graphiques :</u>
- Une forme graphique comporte un polygone, une couleur de remplissage et une couleur de trait.
- Si on détruit la forme graphique, le polygone est détruit aussi.
  Un polygone ne peut faire partie que d'une seule forme graphique.
- Un polygone comporte plusieurs segments.
- Si on détruit le polygone, les segments sont détruits aussi.
  Un segment ne peut faire partie que d'un seul polygone.
- Une palette graphique comporte un ensemble de couleurs
- Une couleur peut appartenir à plusieurs palettes différentes

Exemple diag. de classes



Liens entre classes

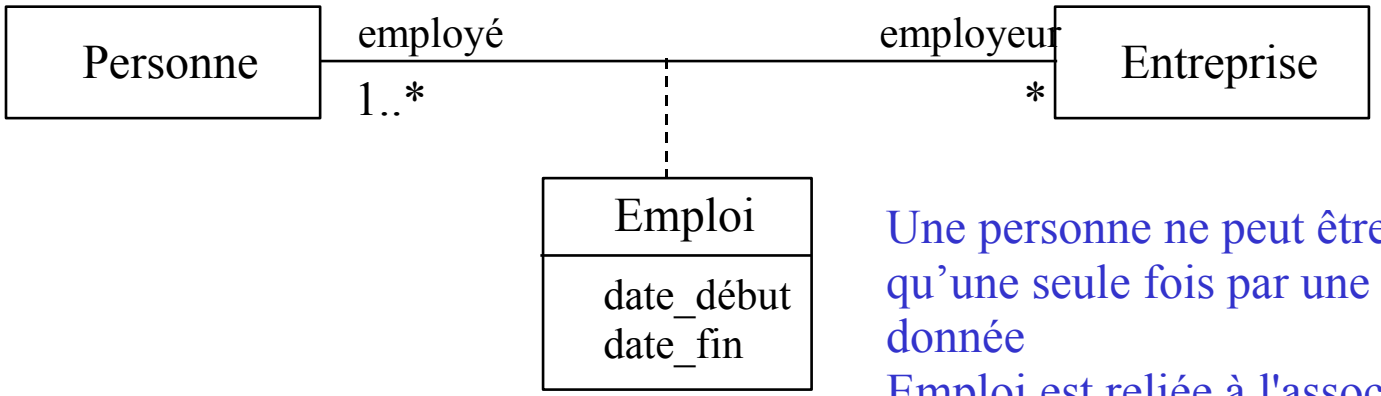
Association n-aire



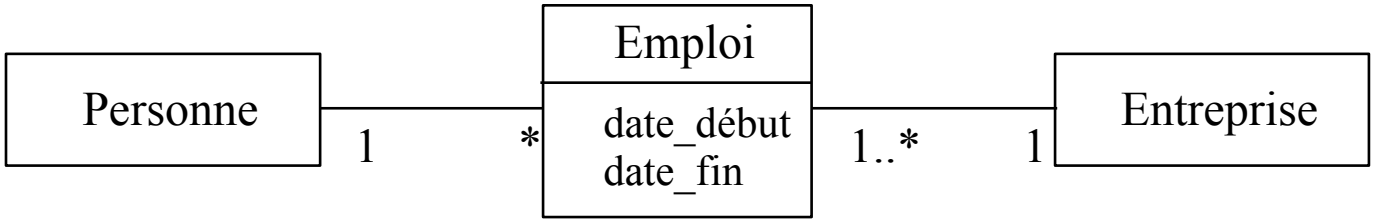
Ex. : enregistrement d'une équipe à chaque saison avec un gardien de but particulier.

Diag. de classes

Quelle est la différence entre ces deux diagrammes ?



Une personne ne peut être employée qu'une seule fois par une entreprise donnée
Emploi est reliée à l'association Personne/Entreprise



Une personne peut avoir plusieurs périodes d'emploi dans une même entreprise

PLUS DE 40 ans D'EXPERIENCE depuis 1967
iut VALENCIENNES CAMBRAI - MAUBEUGE
Informatique DUT INFO

Polymorphisme

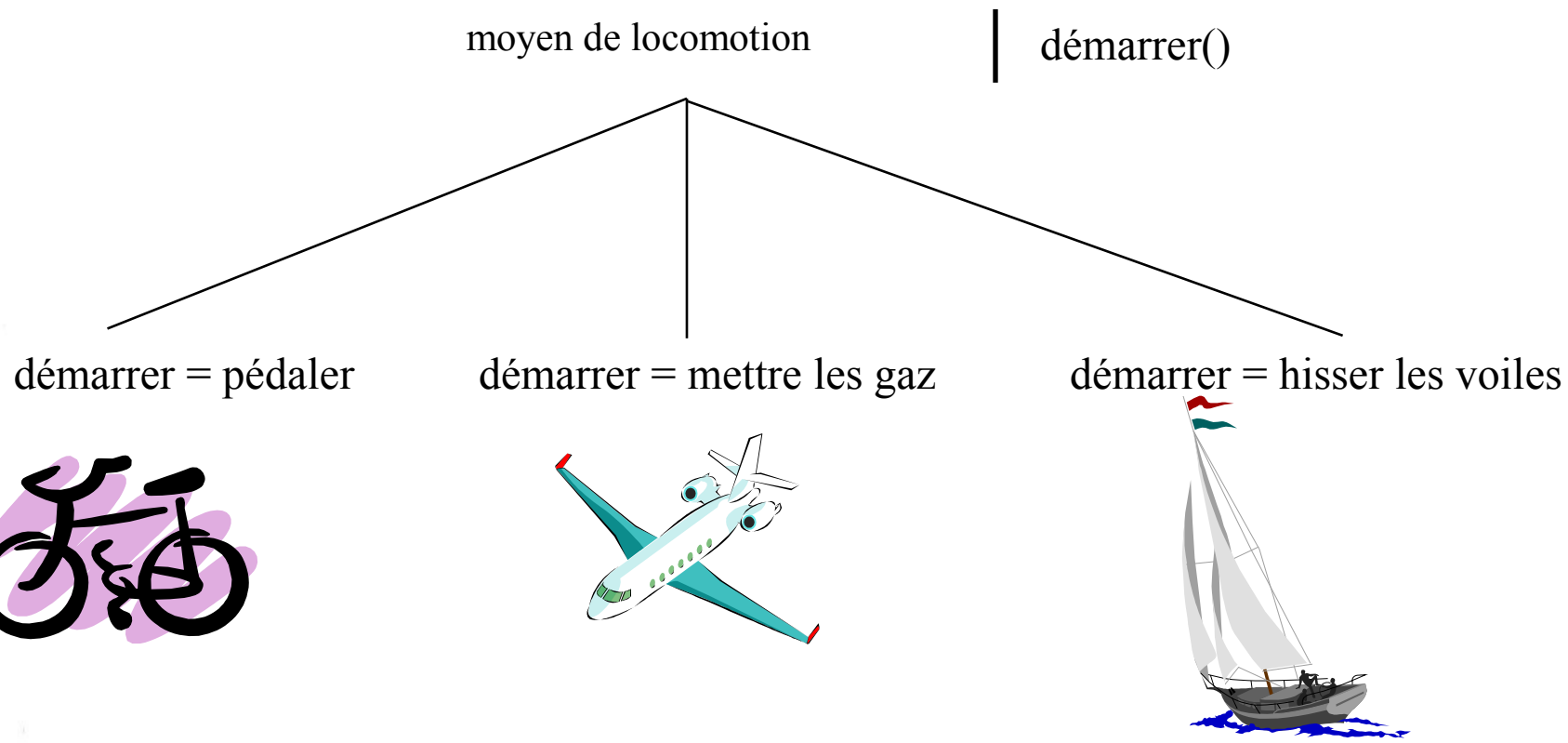
Polymorphisme :

capacité des objets d'une même hiérarchie à répondre différemment à la même demande

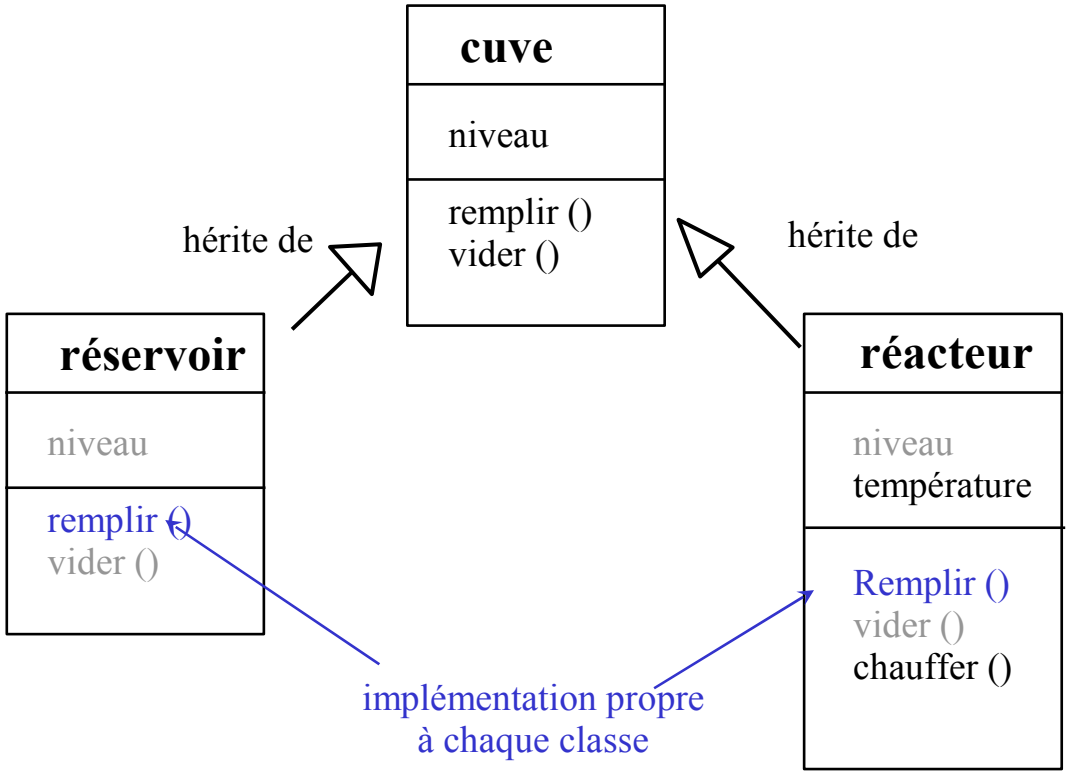
→ une méthode peut être interprétée de différentes façons

IUT

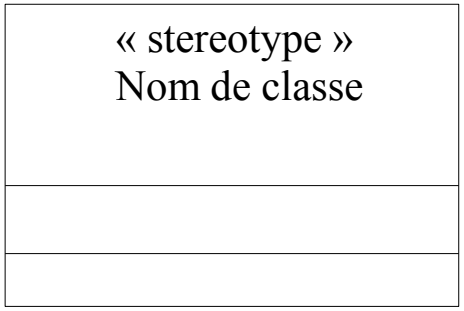
Ex. 1


moyen de locomotion
démarrer()
démarrer = pédaler
démarrer = mettre les gaz
démarrer = hisser les voiles

Ex. 2


cuve
niveau
remplir ()
vider ()
hérite de
hérite de
réservoir
niveau
remplir ()
vider ()
réacteur
niveau
température
Remplir ()
vider ()
chauffer ()
implémentation propre
à chaque classe

Stéréotypes



- Permet d'utiliser des classes prédéfinies tout en leur donnant une signification différente :

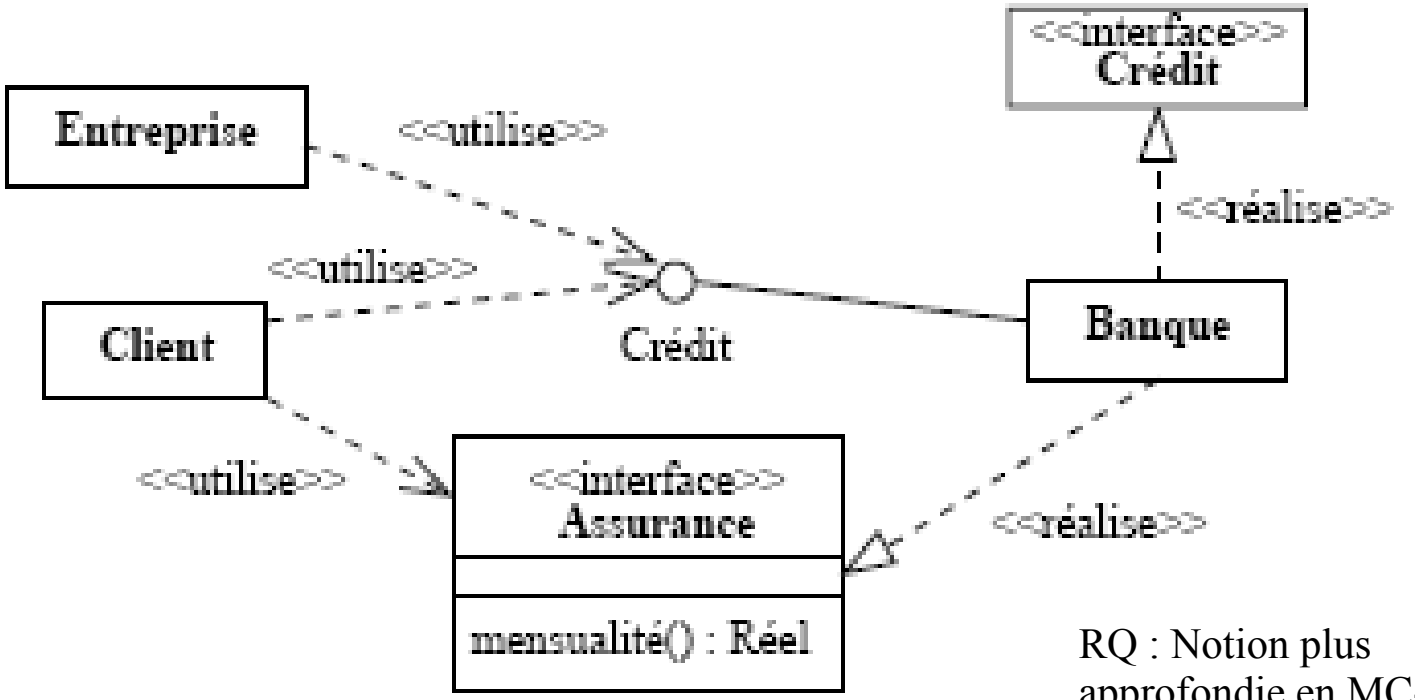
  Si la classe A a le stéréotype B, alors A se comporte comme B , tout en ayant une signification différente

Stéréotypes

- Ex. de stéréotypes :
  - « **enumération** »**:** classe définissant un ensemble d'identificateurs formant le domaine de valeur d'un type.
  - *«* **utilitaire** *» :* classe réduite au concept de module ; ne peut pas être instanciée.
  - *«* **acteur » :** classe modélisant un ensemble de rôles joués par un acteur.
  - *«* **interface » :** classe contenant uniquement une description des opérations visibles.
  - *«* **exception » :** classe modélisant un cas particulier de signal : les exceptions

Ex. de stéréotype : l'interface

- **Interface** = description d'un ensemble d'opérations utilisées pour spécifier un service offert par une classe



RQ : Notion plus approfondie en MC-OMGL1...

Diagramme de classes

description des entités du système et de leurs relations (structure statique)

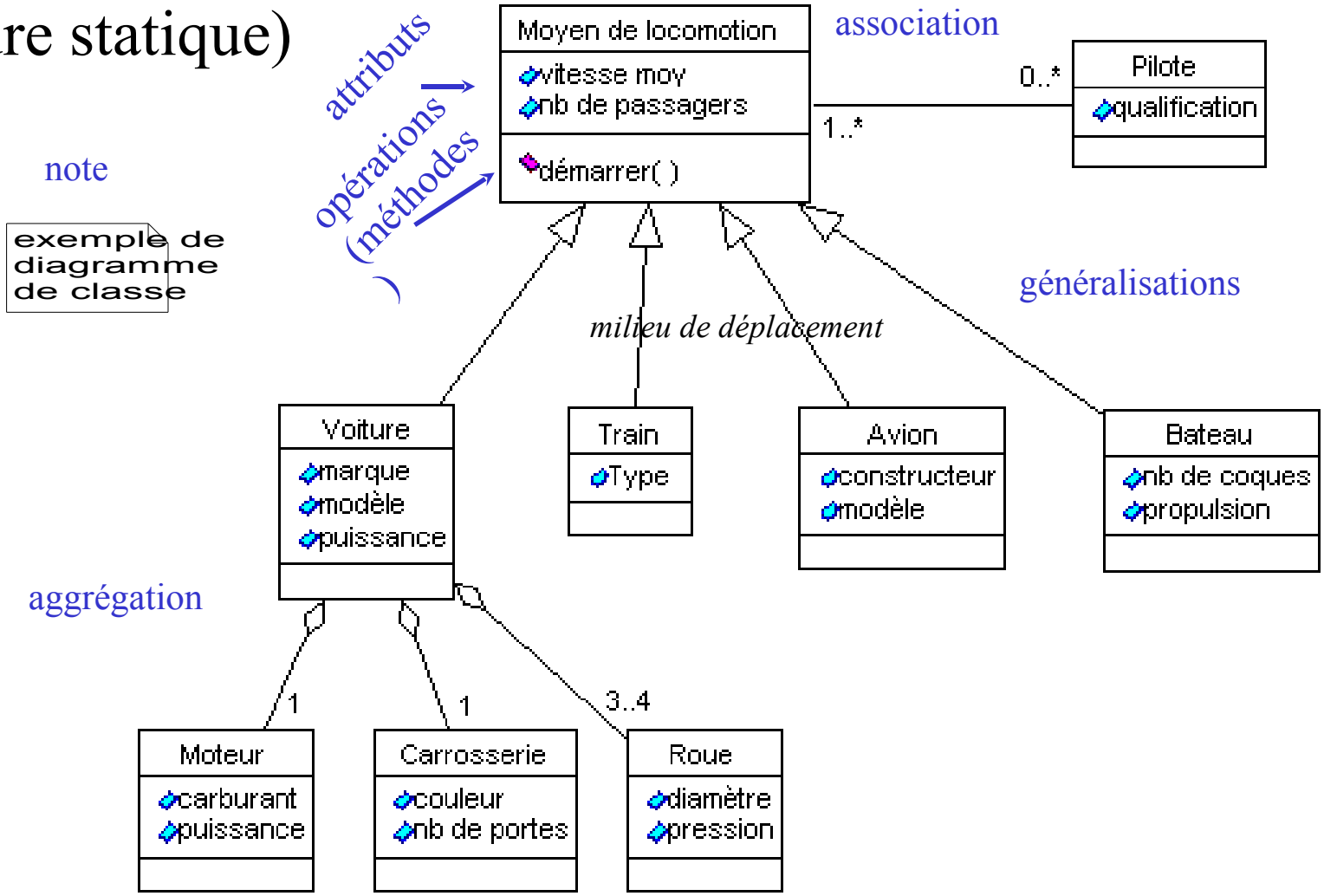


Diagramme de classes

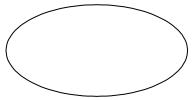
- Se poser des questions dans les cas suivants :
  - Existence d'une classe sans relation
  - Existence d'une classe sans attribut
  - Existence d'une classe sans méthode
  - Existence d'une relation 1-1

→ provient bien souvent d'une erreur dans le modèle...

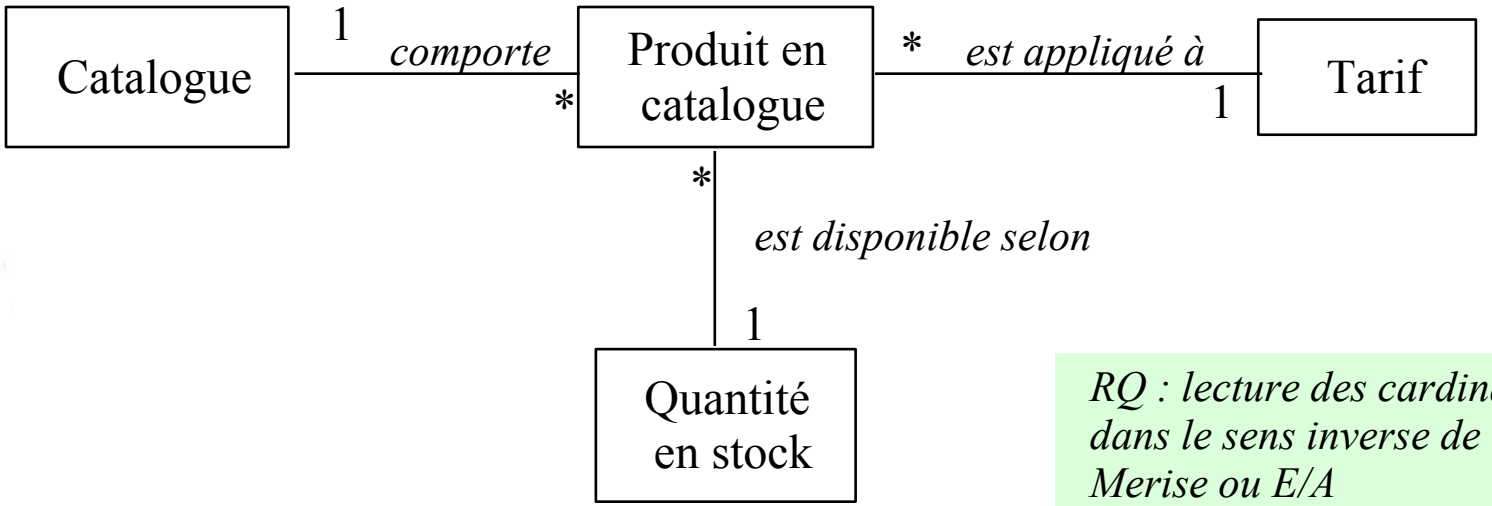
Diagramme d'objets participants

On construit un diagramme d'objet simplifié pour chaque Use Case

ex. de Use Case :



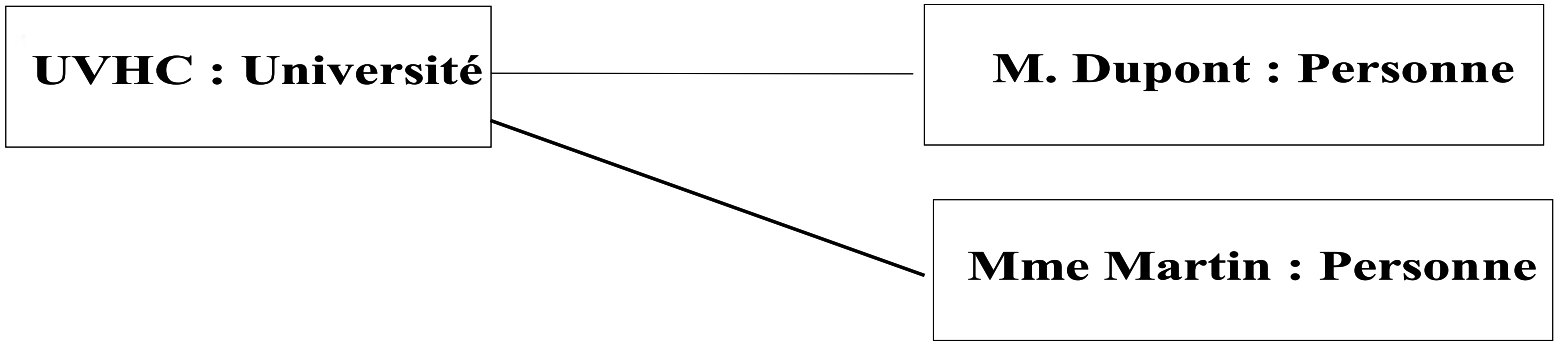
Gestion
disponibilité produits



RQ : lecture des cardinalités dans le sens inverse de celui de Merise ou E/A

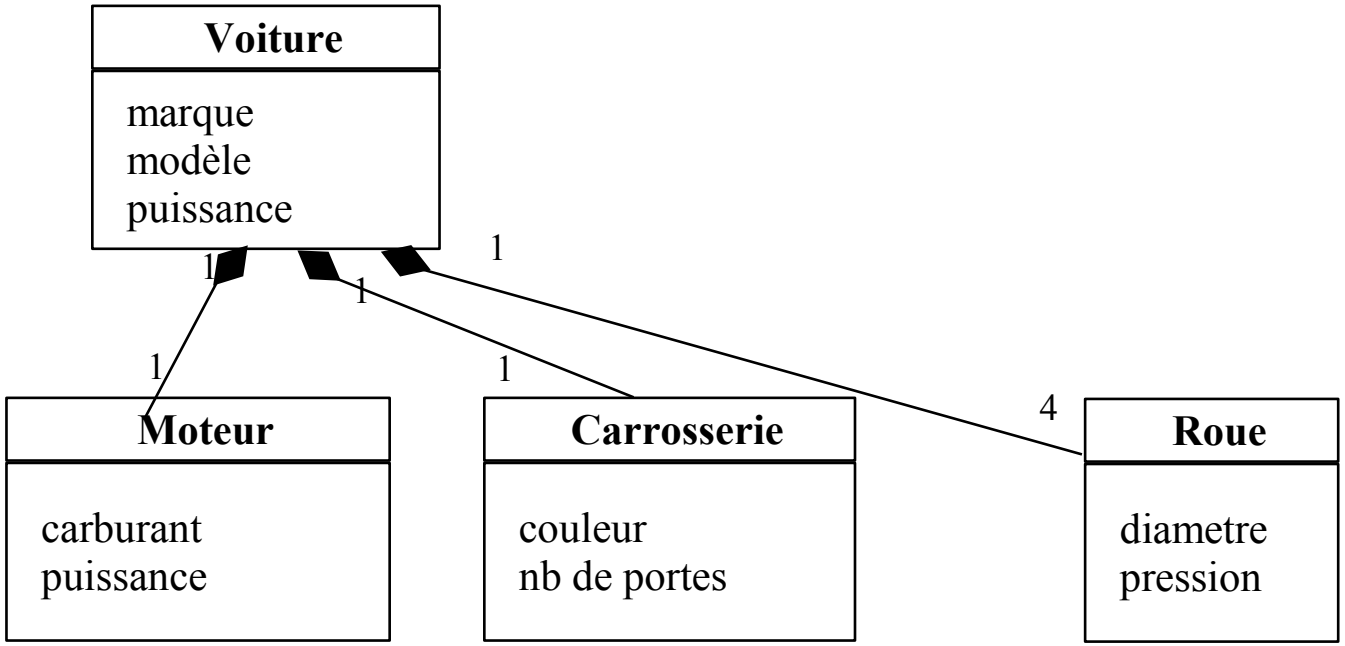
Instance d'association

- Une association entre classes est instanciable : elle devient un lien entre objets



Différence diag. de classes et diag. d'objets

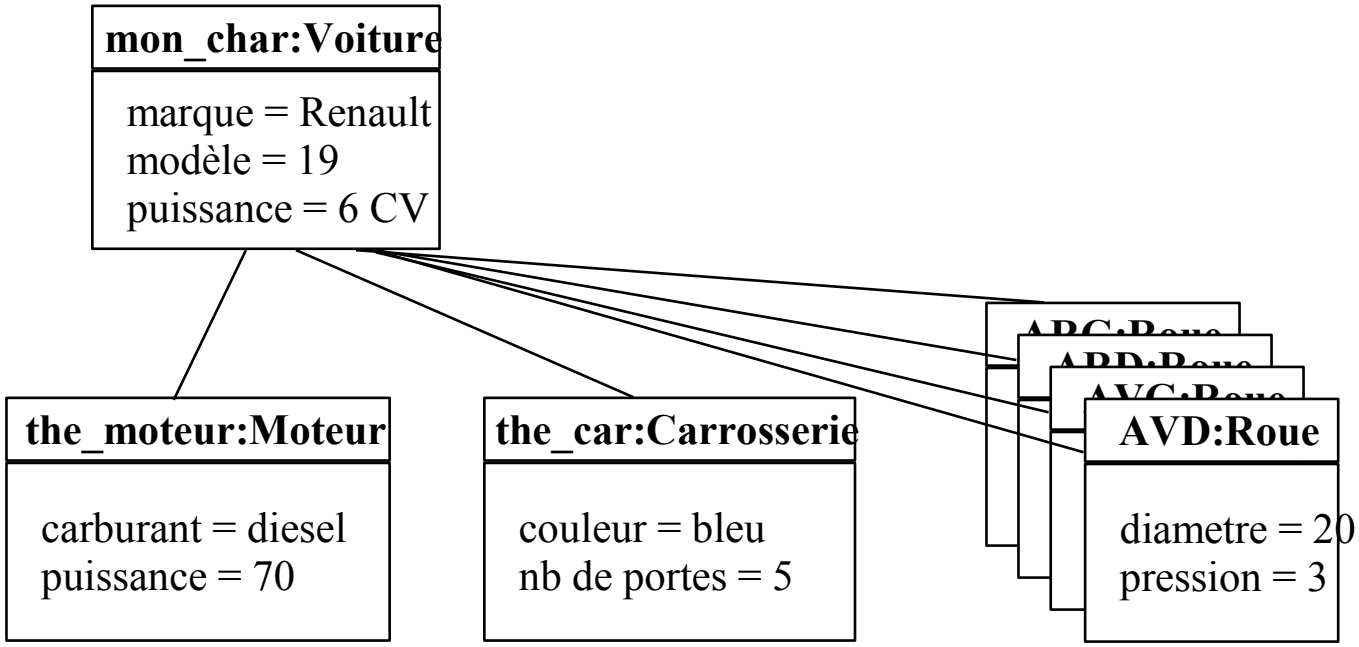
Diagramme de classes



Différence diag. de classes et diag. d'objets

Diagramme d'objets :

description d'instances objets dans un cas particulier



Détail des cas d'utilisation

Pour chaque cas d'utilisation :

- modèle statique :
  - diagramme d'objets participants (simple)
  - identification des objets/classes

    analyse textuelle du cahier des charges :
    - les classes sont très souvent décrites par des noms communs.
    - les opérations sont souvent représentées par des verbes
  - généralisation : ébauche du diagramme de classes
  - rechercher les classes possédant des caractéristiques communes :

    regroupement des classes en paquetages (1 classe appartient à 1 seule catégorie)

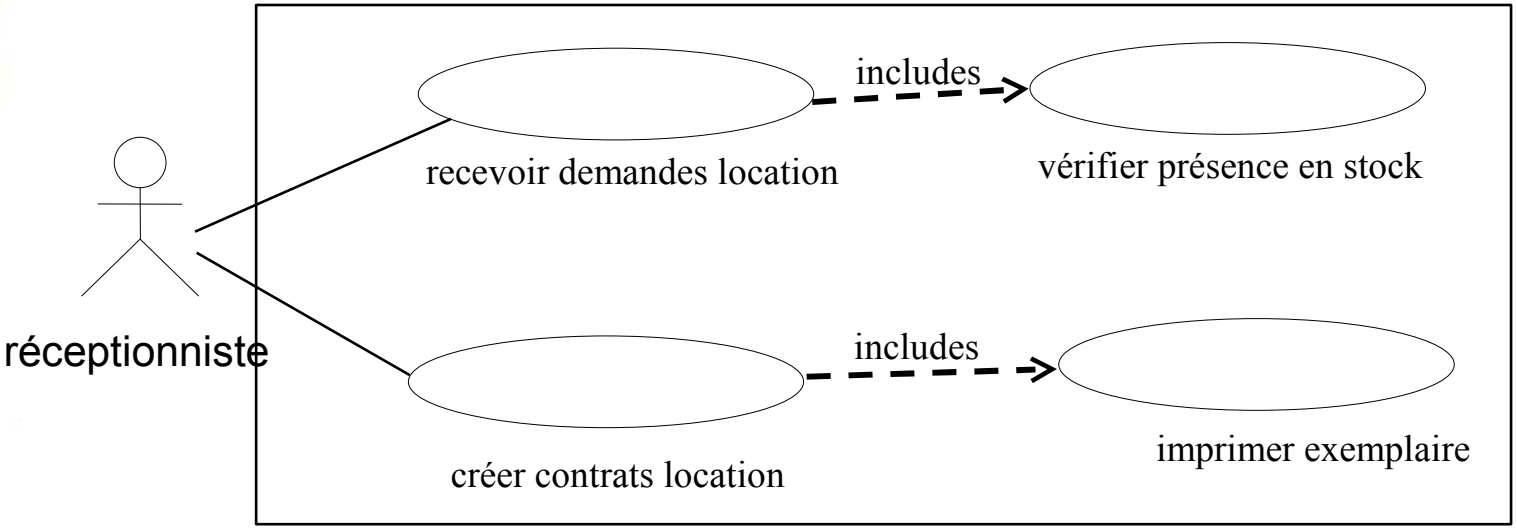
Exemple

L'objectif est de concevoir un système de gestion pour un magasin de location de matériels audio et vidéo.

Le réceptionniste reçoit les demandes de location. Il vérifie la présence du matériel en stock par l'intermédiaire du système. Si l'objet n'est pas en stock, la demande est rejetée. Dans le cas où le matériel est disponible, il demande une pièce d'identité et un chèque de caution au client, le montant de la location et celui de la caution étant fournis par le système.

Le réceptionniste saisit les coordonnées du client, la référence de l'objet ayant été trouvée précédemment par le système lors de sa recherche dans le stock. Le contrat est référencé automatiquement et un exemplaire est imprimé

Ex. : use cases


includes
recevoir demandes location
vérifier présence en stock
réceptionniste
includes
créer contrats location
imprimer exemplaire

Ex : recherche des classes

L'objectif est de concevoir un système de gestion pour un magasin de location de matériels audio et vidéo.

Le réceptionniste reçoit les demandes de location. Il vérifie la présence du matériel en stock par l'intermédiaire du système. Si l'objet n'est pas en stock, la demande est rejetée. Dans le cas où le matériel est disponible, il demande une pièce d'identité et un chèque de caution au client, le montant de la location et celui de la caution étant fournis par le système.

Le réceptionniste saisit les coordonnées du client, la référence de l'objet ayant été trouvée précédemment par le système lors de sa recherche dans le stock. Le contrat est référencé automatiquement et un exemplaire est imprimé

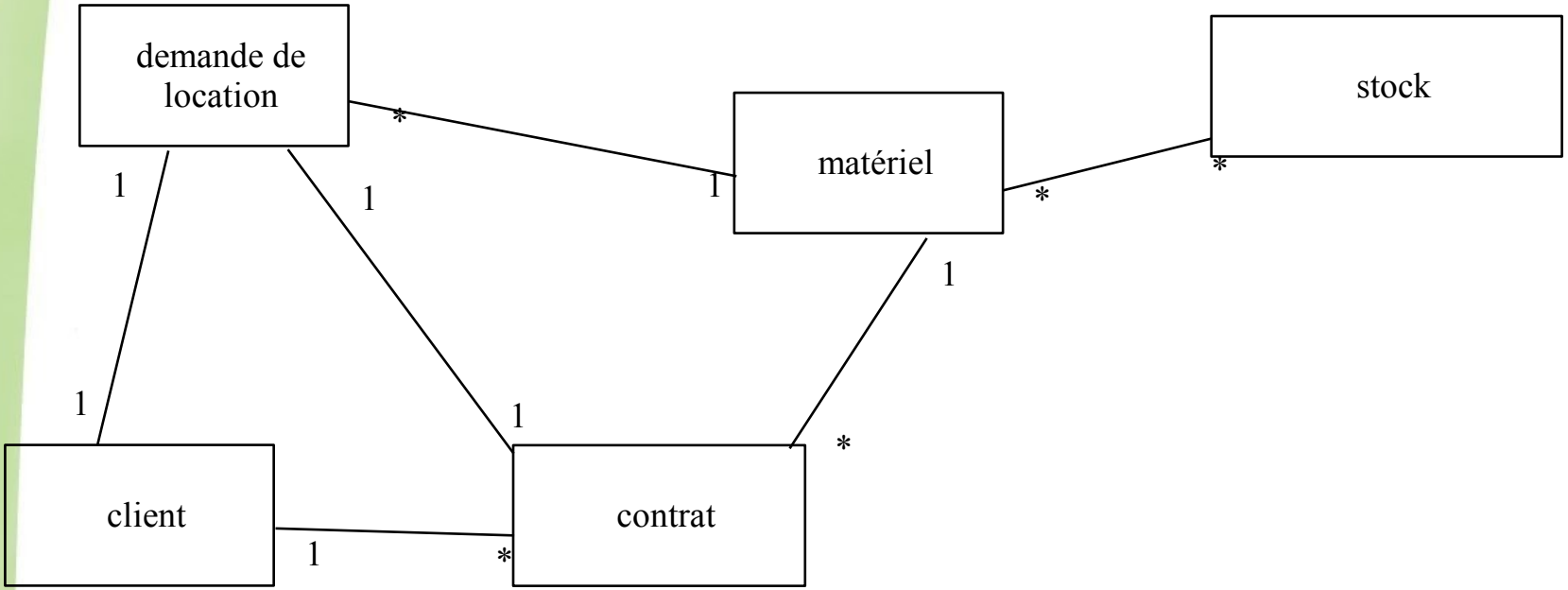
Ex : recherche des méthodes

L'objectif est de concevoir un système de gestion pour un magasin de location de matériels audio et vidéo.

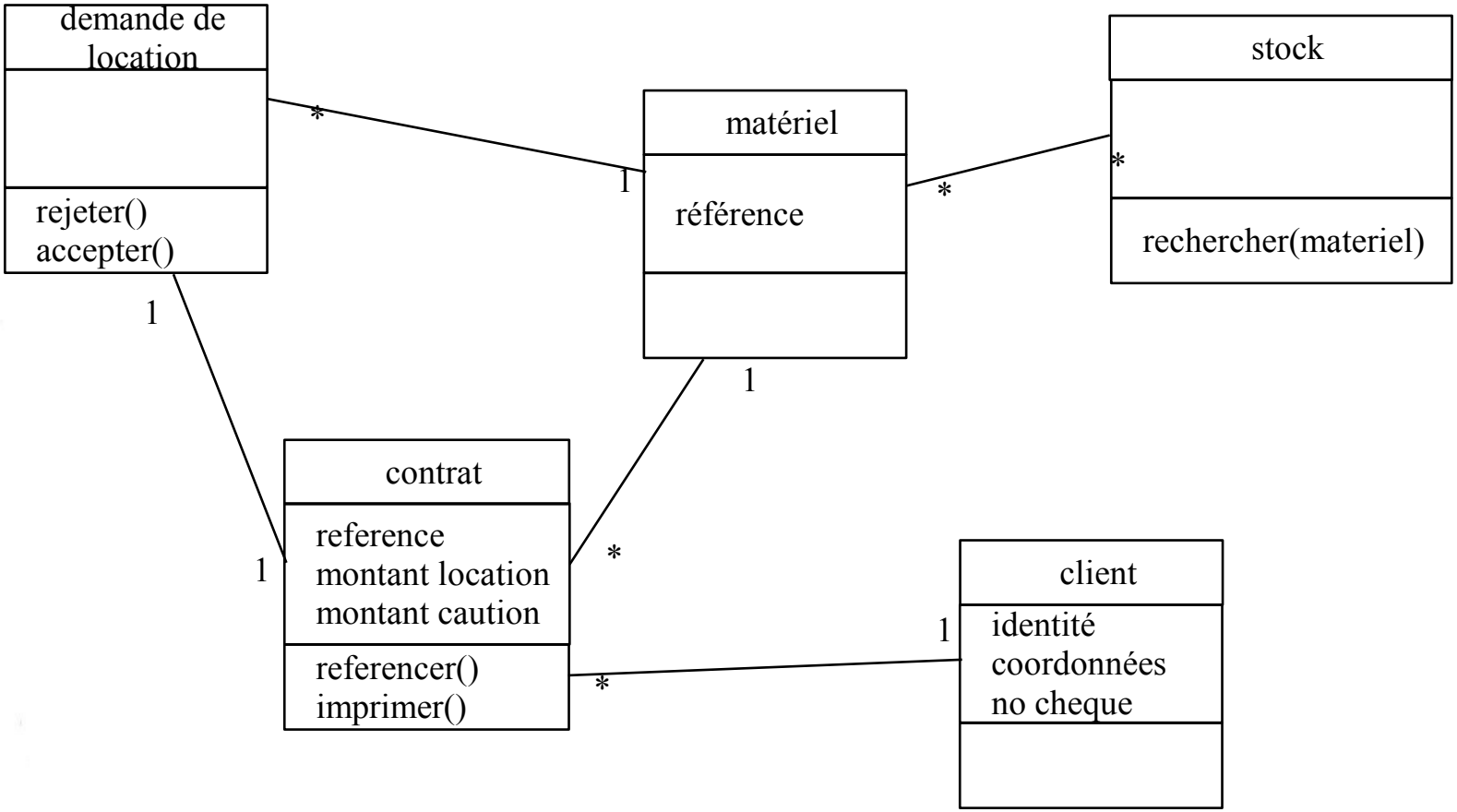
Le réceptionniste reçoit les demandes de location. Il vérifie la présence du matériel en stock par l'intermédiaire du système. Si l'objet n'est pas en stock, la demande est rejetée. Dans le cas où le matériel est disponible, il demande une pièce d'identité et un chèque de caution au client, le montant de la location et celui de la caution étant fournis par le système.

Le réceptionniste saisit les coordonnées du client, la référence de l'objet ayant été trouvée précédemment par le système lors de sa recherche dans le stock. Le contrat est référencé automatiquement et un exemplaire est imprimé

Ex : 1er diagramme

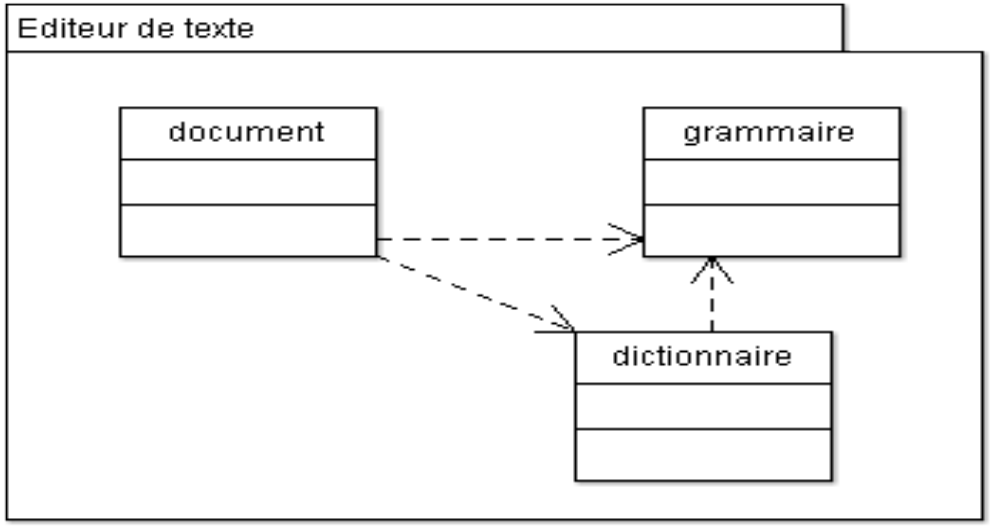


Ex : ébauche du diag. de classes



Identification des paquetages

paquetage (*package*) = regroupement d'éléments du modèle.



Identification des paquetages